Progress®



Use OpenEdge Adapter for Sonic ESB

OpenEdge®

Table of Contents

Learn About OpenEdge Adapter for Sonic ESB....9

What is OpenEdge Adapter for Sonic ESB?....10
Native Invocation methodology....10
Declarative vs. non-declarative approach....10
Annotations....11
.esboe files....11
Web Service Invocation methodology....12
WSM files....12
WSDL files....12
Sonic Management Console....13
Sonic ESB container....14
SonicMQ container....15
SonicMQ Broker....16
Learn about Sonic ESB adapter for PAS for OpenEdge....17
OpenEdge services: usage and management....19

Install OpenEdge Adapter for Sonic ESB....21

Install the OpenEdge ESB Adapter interactively....21
Install the OpenEdge ESB adapter on UNIX....25
Install the OpenEdge ESB Adapter in silent mode....31

Guidelines for Using and Programming for the OpenEdge Adapter for Sonic ESB....33

Native Invocation methodology....33
Create an invocation file....34
Declarative invocation files....35
Non-declarative invocation files....40
Import Native Invocation files into Sonic Workbench....41
ABL parameter naming....43
Test an ESB process containing OpenEdge native invocation services....50
Configure Progress Developer Studio for OpenEdge for ESB annotations....51
Default ESB annotations....51
Generate ESB invocation files....52
Web Services Invocation methodology....53
Differences between the Web Services Invocation methodology in the OpenEdge Adapter for Sonic ESB and the WSA....54
Service definition considerations....55

Session models....55
Sonic message handling run-time parameters....56

Run a sample Sonic ESB Process using Native Invocation....59

Develop ABL source....60
Configure OpenEdge servers....60
Annotate the source with Progress Developer Studio for OpenEdge....61
Build Native Invocation and r-code files....64
Prerequisites to start Sonic Workbench....65
Create an ESB process in Sonic Workbench....65
Update the Sonic Management Console....71
Test the ESB process....71

Copyright

© 2021 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Defrag This, Deliver More Than Expected, DevReach (and design), Icenium, Inspec, Ipswitch, iMacros, Kendo UI, Kinvey, MessageWay, MOVEit, NativeChat, NativeScript, OpenEdge, Powered by Chef, Powered by Progress, Progress, Progress Software Developers Network, SequeLink, Sitefinity (and Design), Sitefinity, Sitefinity (and design), SpeedScript, Stylus Studio, Stylized Design (Arrow/3D Box logo), Styleized Design (C Chef logo), Stylized Design of Samurai, TeamPulse, Telerik, Telerik (and design), Test Studio, WebSpeed, WhatsConfigured, WhatsConnected, WhatsUp, and WS_FTP are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries.

Analytics360, AppServer, BusinessEdge, Chef Automate, Chef Compliance, Chef Desktop, Chef Habitat, Chef WorkStation, Corticon.js, Corticon Rules, Data Access, DataDirect Autonomous REST Connector, DataDirect Spy, DevCraft, Fiddler, Fiddler Everywhere, FiddlerCap, FiddlerCore, FiddlerScript, Hybrid Data Pipeline, iMail, JustAssembly, JustDecompile, JustMock, KendoReact, NativeScript Sidekick, OpenAccess, PASOE, Pro2, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, Push Jobs, SafeSpaceVR, Sitefinity Cloud, Sitefinity CMS, Sitefinity Digital Experience Cloud, Sitefinity Feather, Sitefinity Insight, Sitefinity Thunder, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, Supermarket, SupportLink, Unite UX, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

July 2021

Product version: Progress OpenEdge 12.4

Preface

Purpose

The *Use OpenEdge Adapter for Sonic ESB* guide describes how OpenEdge Adapter for Sonic ESB® allows OpenEdge ABL procedures to be incorporated into ESB processes on the Sonic Enterprise Service Bus. The ABL procedures are accessed through Sonic ESB either directly, or as a web service.

Audience

This guide is aimed at ABL developers who are experienced in writing ABL classes and procedures and are expected the work with OpenEdge Adapter for Sonic ESB.

Organization

This book is organized into the following chapters:

- **Learn About OpenEdge Adapter for Sonic ESB**—Explains OpenEdge Adapter for Sonic ESB and how to start working with it
- **Install OpenEdge Adapter for Sonic ESB**—Explains the different types of installation processes
- **Guidelines for Using and Programming for the OpenEdge Adapter for Sonic ESB**— Discusses certain general considerations related to OpenEdge Sonic ESB services and the operation of the OpenEdge Adapter for Sonic ESB
- **Run a sample Sonic ESB Process using Native Invocation**—Explains the steps of a sample Sonic ESB Process using Native Invocation

Documentation conventions

See Documentation Conventions for an explanation of the terminology, format, and typographical conventions used throughout the OpenEdge content library.

1

Learn About OpenEdge Adapter for Sonic ESB

OpenEdge® provides the OpenEdge Adapter for Sonic ESB® to allow OpenEdge ABL procedures to be incorporated into ESB processes on the Sonic Enterprise Service Bus. ABL procedures are accessed through Sonic ESB either directly when using the Native Invocation methodology, or as a Web Service when using the web service Invocation methodology.

This topic describes the architecture and where to find further information about using these adapter products.

Note: The **Progress SonicMQ** and **Sonic ESB** products were acquired by **Aurea Software** and became part of the Aurea CX Platform, which was subsequently renamed to **Aurea Integration Solutions**. In line with this, Sonic ESB became Aurea CX Messenger ESB, and then was renamed to **Aurea Messenger**.

For details, see the following topics:

- What is OpenEdge Adapter for Sonic ESB?
- Native Invocation methodology
- Web Service Invocation methodology
- Sonic Management Console
- Sonic ESB container
- SonicMQ container
- SonicMQ Broker
- Learn about Sonic ESB adapter for PAS for OpenEdge

- OpenEdge services: usage and management

What is OpenEdge Adapter for Sonic ESB?

Sonic ESB is an industry-leading application integration framework that provides exceptional performance, reliability, and security. Its service-based architecture supports the deployment of discrete applications (services) that exchange messages according to sophisticated automated workflow processes. It also supports the exposure of deployed applications as standard web services that can process requests sent through HTTP or HTTPS (HTTP/S) from any compliant client application external to the Sonic ESB.

The OpenEdge Adapter for Sonic ESB enables PAS for OpenEdge applications to participate as services on the Enterprise Service Bus (ESB) and take part in workflow processes managed by the Sonic ESB. The OpenEdge Adapter for Sonic ESB supports two distinct methodologies:

- Native Invocation methodology
- Web Service Invocation methodology

Native Invocation methodology

When a Sonic ESB process calls an OpenEdge service using the Native Invocation methodology, ABL procedures are called directly through an OpenAPI call to PAS for OpenEdge. The Native Invocation methodology depends on an invocation (`.esboe`) file. An `.esboe` file is typically based on annotations within your ABL source, but can also be generated from the Proxy Generator (ProxyGen utility), and is imported directly into Sonic Workbench.

The Native Invocation methodology provides the following benefits:

- Simplified exposure of ABL code as a service
- Simplified process of mapping ABL parameters to Sonic messages
- Reduced overhead through improving run-time performance over the Web Service Invocation methodology by eliminating conversions to and from SOAP messages

Declarative vs. non-declarative approach

OpenEdge developers have the choice of two approaches for exposing ABL procedures and functions for the Native Invocation methodology:

- **Declarative**—Captures information about publicly exposable procedures in the source code through the use of annotations.
- **Non-declarative**—Captures information about publicly exposable procedures through the use of a utility such as ProxyGen.

The declarative approach is recommended to OpenEdge developers as a best practice. When following the declarative approach, developers enter relevant information for making a procedure, function, or external procedure publicly exposable when the source code is written. This information is then stored with the source code and during the build process the source code is captured as an invocation (`.esboe`) file. If the developer does not want to capture information about publicly exposable procedures with the source code, then you can use the non-declarative approach.

Annotations

Annotations are a way to capture information in the source code that extends the ABL language syntax. Validation is available to confirm that the information entered adheres to the predefined syntax and format.

At a minimum, the required annotation is simply a declaration that the ABL procedure is to be publicly available. The best practice is to add annotations to the ABL source code through the Progress Developer Studio for OpenEdge Annotation wizard. However, annotations can be added to the source code with whatever the developer uses to create the source code.

.esboe files

The Native Invocation methodology of the OpenEdge Adapter for Sonic ESB depends on an invocation file for the easy integration of services in the Sonic ESB business process through itineraries. For ABL procedures and functions, this invocation file is the `.esboe` file. An `.esboe` file supplies the information necessary for Sonic to make a call to PAS for OpenEdge to execute a single ABL procedure. After it is created, the `.esboe` file can be dropped onto the canvas of the Sonic ESB Process Editor in Sonic Workbench to create a new step in an ESB process. For more information about generating an ESB process, see your Sonic documentation.

Creation of the invocation file depends on the developer's choice of a declarative or non-declarative approach to development. The following table describes the options.

Source code development approach (declarative/ non-declarative)	Source code development tool	.esboe file generation
Declarative (annotated code)	Progress Developer Studio for OpenEdge	• Automatically generated when the source is saved to disk and when the project has **Build Automatically** selected • Automatically generated when **Clean** is selected for the project and when **Build Automatically** is selected • Automatically generated when the source is compiled
	Text editor or a source code generator other than Progress Developer Studio for OpenEdge	Run the ProxyGen utility using compiled source code.
Non-declarative (code not annotated)	Any	Run ProxyGen using compiled source code.

Web Service Invocation methodology

When employing the Web Service Invocation methodology, OpenEdge services are transformed into OpenEdge web services that are deployed (installed) into a Sonic ESB container. At run time, the OpenEdge Adapter for Sonic ESB receives SOAP messages from the SonicMQ Broker, converts them to ABL, and sends them to PAS for OpenEdge to be processed. The adapter then receives ABL messages from PAS for OpenEdge, converts them to SOAP, and returns them to the broker, where they are then routed according to a Sonic ESB process itinerary or a web service reply mechanism.

For more information about OpenEdge web services and how they are created, see *Develop ABL Services*.

In addition, for each OpenEdge service, the OpenEdge Adapter for Sonic ESB processes the WSM (or WSD) file.

WSM files

The Web Service Mapping (WSM) file contains the XML-encoded instructions that enable the OpenEdge Adapter for Sonic ESB to process SOAP messages according to the needs of the OpenEdge application. A WSM file is created for each service using the OpenEdge Open Client Proxy Generator utility (ProxyGen).

For more information about ProxyGen and how to create WSM files, see *Use the Open Client Toolkit*.

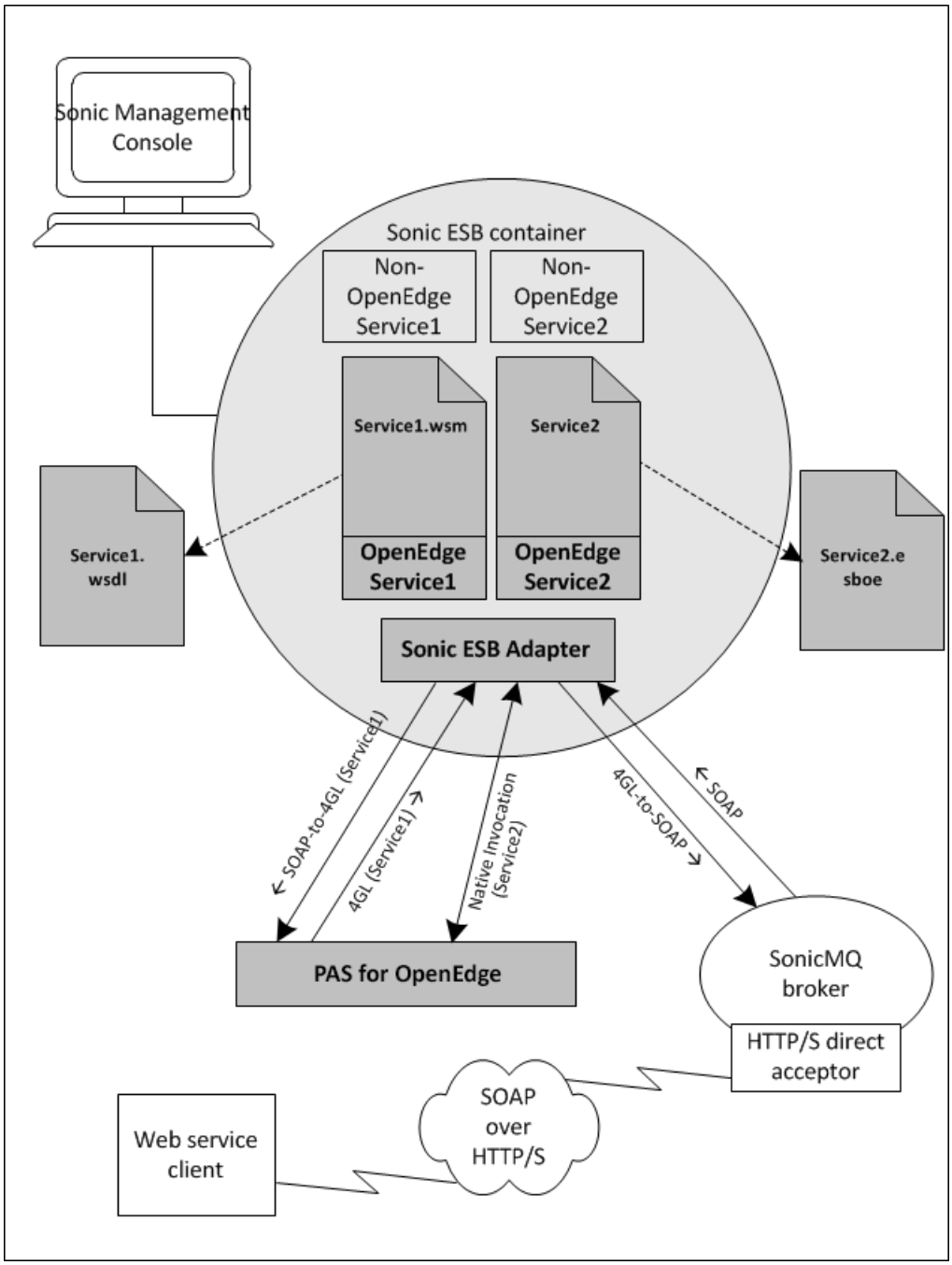
In the process of installing an OpenEdge service in the Sonic ESB environment, the OpenEdge Adapter for Sonic ESB takes the WSM file as its primary input.

WSDL files

The Web Services Description Language (WSDL) file is an optional XML file that is generated by the ProxyGen utility and used by the Sonic Web Services Invocation framework to generate a proper SOAP message. The files contain information that is required by programmers of client applications that call the service. The files are standard components for developing of web service clients. However, the WSDL file generated by the ProxyGen utility, is specifically for use with Sonic's Web Service Invocation and is not appropriate for use with outside clients. WSDL files are stored as resources in Sonic ESB and can be viewed with Sonic Workbench.

For more information about the WSDL files generated for the OpenEdge Adapter for Sonic ESB and how web service clients must use them, see "Web Service client details" in *Learn about Open Clients*.

The following figure is a simplified illustration of the function of the OpenEdge Adapter for Sonic ESB in the context of Sonic ESB. The components and functions in this figure are explained in more detail in the topics that follow.



Sonic Management Console

The Sonic Management Console is the graphical user interface to a set of tools for managing the entire Sonic messaging and application environment, including both Sonic ESB and the required underlying messaging infrastructure, SonicMQ. The Sonic Management Console is the primary way to perform administration and configuration tasks for all services and the elements (such as process definitions) and resources (such as service definitions) that support them.

A command-line tool, called the ESB Admin Tool, can also be used for most of these administrative tasks.

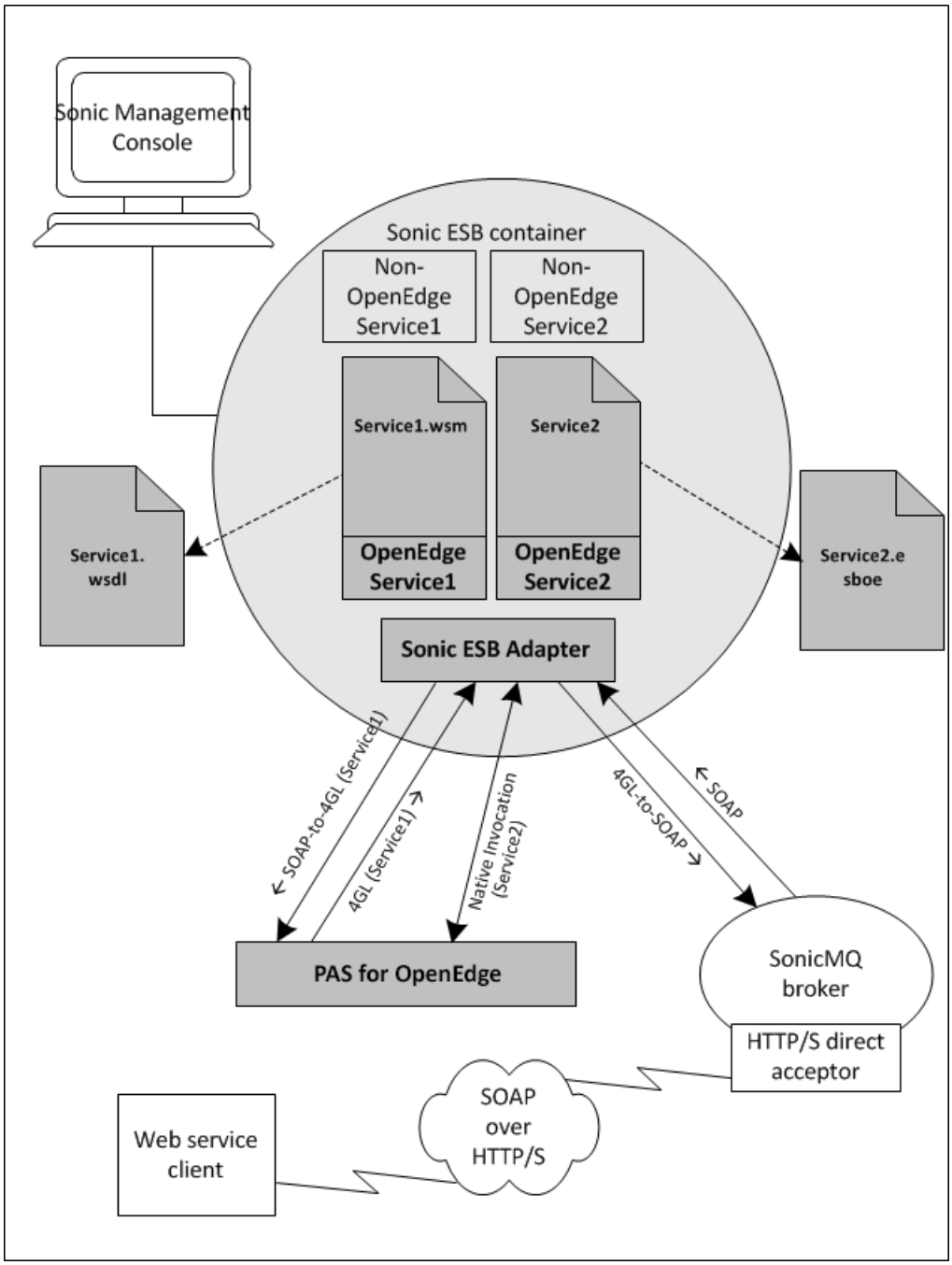
Note: The OpenEdge Adapter for Sonic ESB does not appear in the Sonic Management Console. The Sonic Management Console is the presence of the two OpenEdge Services entries (**OpenEdge Native Services** and **OpenEdge Web Services**) in the **ESB Configured Objects** that indicates that the adapter is installed. Furthermore, the OpenEdge Adapter for Sonic ESB does not appear in the user interface of any OpenEdge tool.

For more information about these Sonic ESB management tools, see the Sonic ESB product documentation.

Sonic ESB container

All Sonic ESB services are deployed to a Sonic ESB container. The container hosts various components, including ESB processes, in addition to services. It also provides a number of support functions.

The following figure shows two OpenEdge services supported by the OpenEdge Adapter for Sonic ESB and two non-OpenEdge services. For Service1, OpenEdge has an associated WSM file and an associated WSDL file, and uses the Web Service Invocation methodology. For Service2, OpenEdge has an associated `.esboe` file, and uses the Native Invocation methodology.



For more information about the Sonic ESB container, see the Sonic ESB product documentation.

SonicMQ container

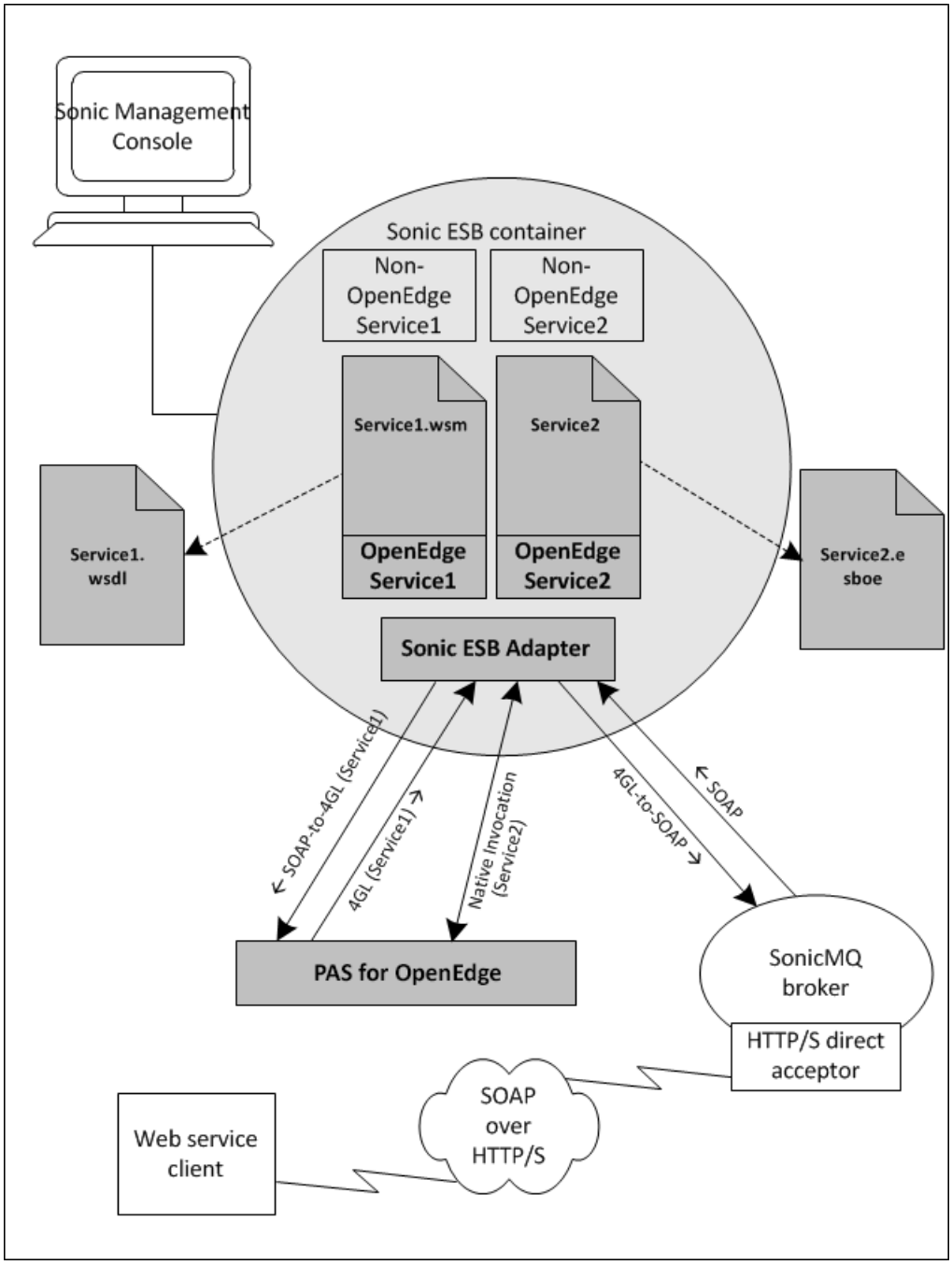
SonicMQ is the messaging backbone on which Sonic ESB relies for communications. Each Sonic ESB container must be deployed to a SonicMQ container.

For more information about the SonicMQ container, see the SonicMQ product documentation.

SonicMQ Broker

The SonicMQ messaging system is based on one or more brokers that establish connections with clients, perform routing functions, and implement security measures. Every client of SonicMQ has a persistent connection to a SonicMQ Broker.

As shown, a SonicMQ Broker can be configured with an HTTP/S acceptor. In effect, such an acceptor lets the broker function as a web server, enabling Sonic ESB to expose its services as standard web services that process requests from external clients using the internet.

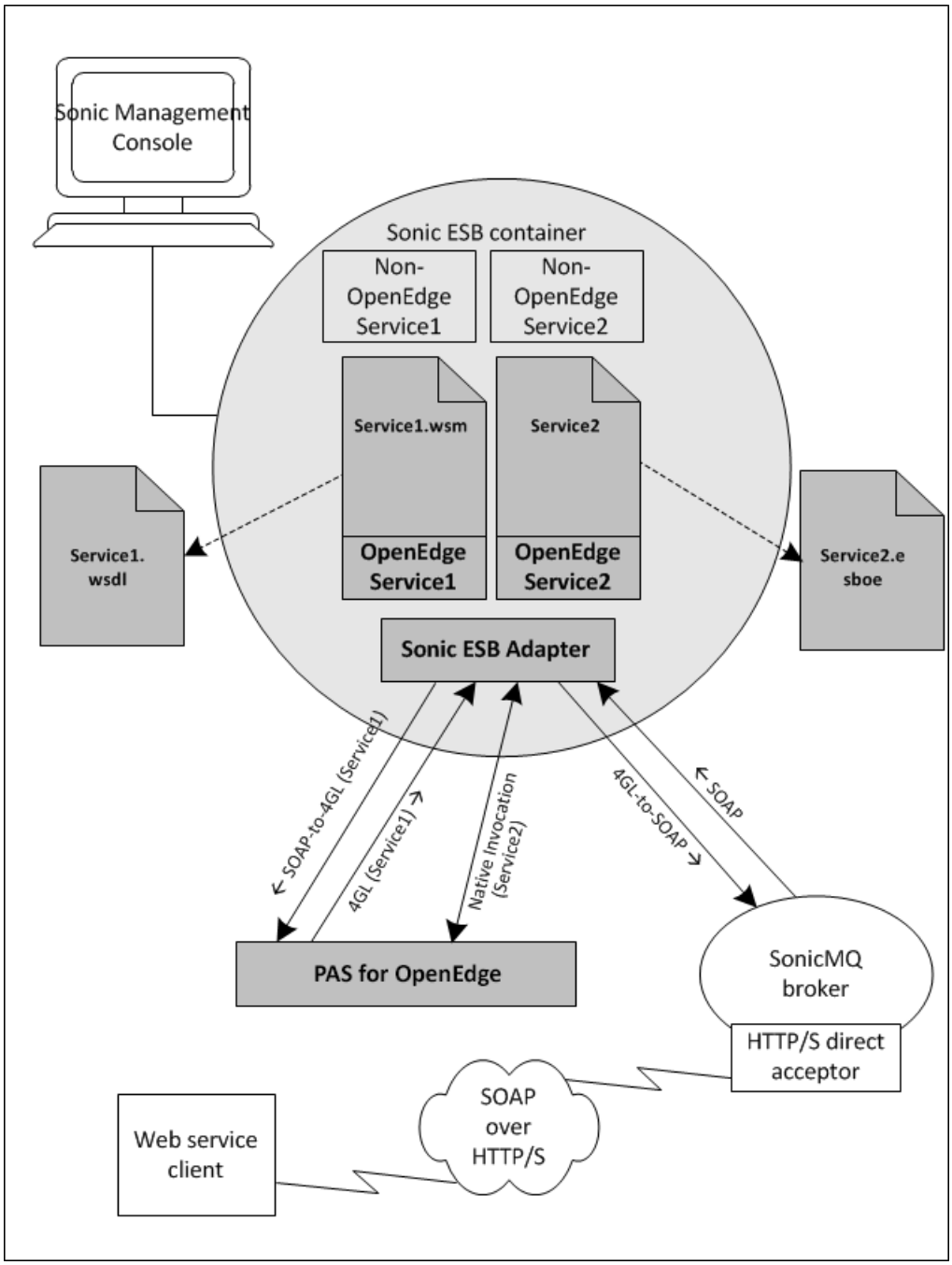


For more information about the SonicMQ Broker, see the SonicMQ product documentation.

Learn about Sonic ESB adapter for PAS for OpenEdge

PAS for OpenEdge is the OpenEdge product for the storage and execution of business logic, including OpenEdge services executed by Sonic ESB. For each client request to a service, PAS for OpenEdge executes the application code, processing the input from the client and returning results and errors accordingly.

The following figure represents PAS for OpenEdge as a single entity. This is a simplification of the server architecture. PAS for OpenEdge launches PAS for OpenEdge agents, which run as separate processes controlled by the PAS for OpenEdge broker and perform the run-time functions.



Note that the connection between the OpenEdge Adapter for Sonic ESB and PAS for OpenEdge can be tunneled through an OpenEdge implementation of the Transport Layer Security (TLS) to provide PAS for OpenEdge intranet connection security in addition to any security managed by Sonic ESB.

For more information about PAS for OpenEdge, see *Learn about Progress Application Server for OpenEdge*. For more information about secure connections to PAS for OpenEdge, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

OpenEdge services: usage and management

With Sonic ESB, an OpenEdge service can be accessed directly as a web service by an industry-standard web service client, or it can participate in workflow processes managed by Sonic ESB. The functions of the OpenEdge service are managed and made accessible entirely within the Sonic ESB environment.

2

Install OpenEdge Adapter for Sonic ESB

To install the OpenEdge Enterprise Service Bus (ESB) adapter on Windows, download the OpenEdge ESB adapter software image from the Progress Download Center, and launch the interactive installation program. The software is provided in a ZIP fie. You must extract the contents of the ZIP file to your machine.

Note that during installation, you can optionally capture a `response.ini` file. You can use the `response.ini` file to create a silent installation script, which automates the installation of the OpenEdge ESB adapter across a network.

Note: The Sonic ESB installer package includes the sonic `.jar` files by default.

For details, see the following topics:

- Install the OpenEdge ESB Adapter interactively
- Install the OpenEdge ESB adapter on UNIX
- Install the OpenEdge ESB Adapter in silent mode

Install the OpenEdge ESB Adapter interactively

To launch the OpenEdge ESB adapter, you must have administrator privileges on the machine where you are installing the OpenEdge ESB adapter. For information about planning your installation, see *Install OpenEdge*.

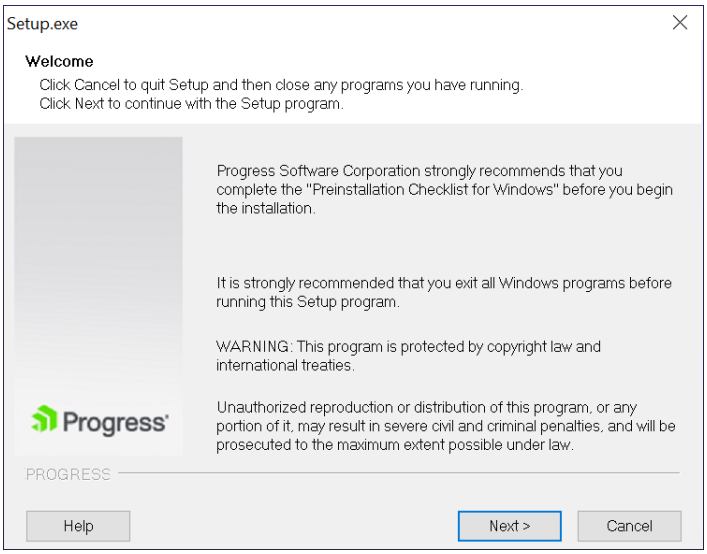
To install the OpenEdge ESB adapter on your machine, complete the following steps:

1. Close all applications.

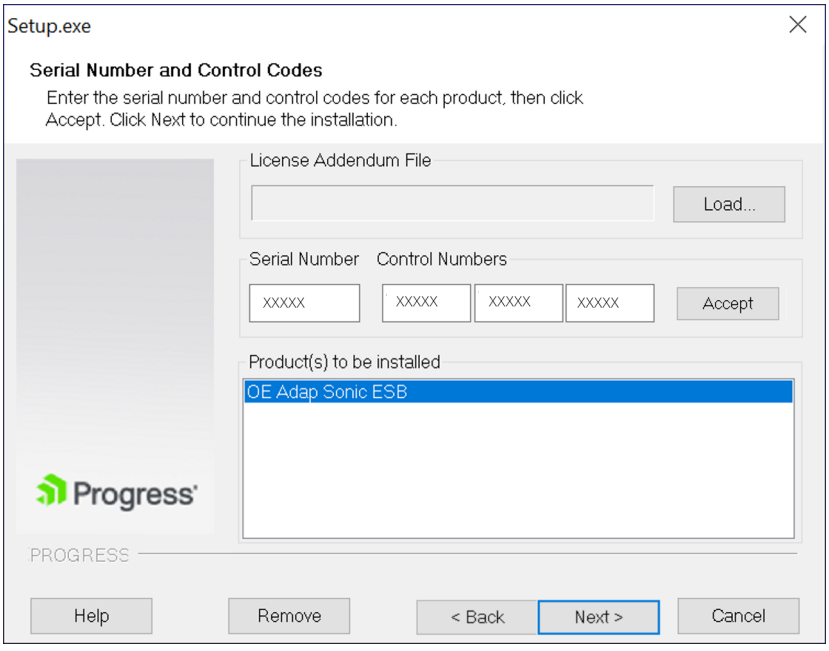
   Running applications or tasks may interfere with the installation because they could use files that OpenEdge needs to complete the installation. Stop any processes that are running, and close any files that are located in the same directory where you want to install the OpenEdge ESB adapter.

2. Double-click the `setup.exe` file.

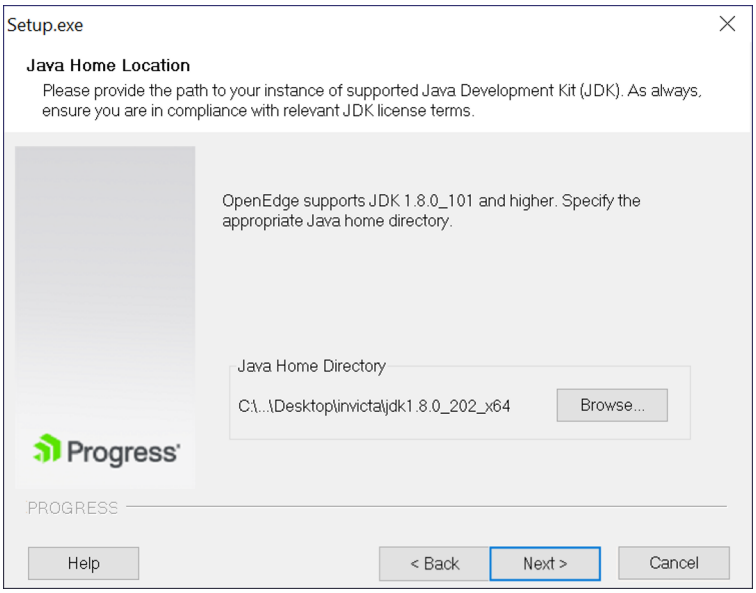
3. Read the information in the **Welcome** dialog box, verify that all the other applications are closed, and click **Next**.



4. In the **Serial Number and Control Codes** dialog box, enter the serial number and control number for the OpenEdge ESB adapter and click **Accept**. Select **OE Adap Sonic ESB** from the **Product(s) to be installed** box and click **Next**.



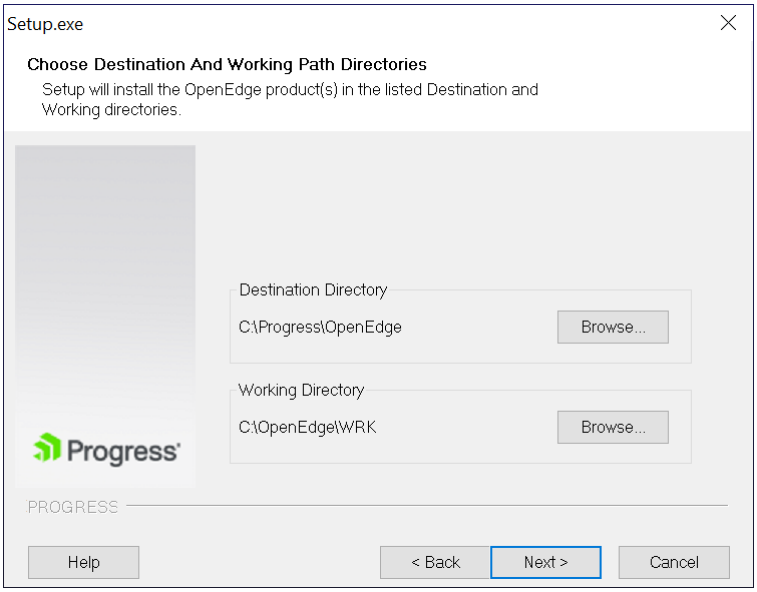
5. In the **Java Home Location** dialog box, click **Browse**, and find the location of the JDK installation for the Java Home Directory. Click **Next**.



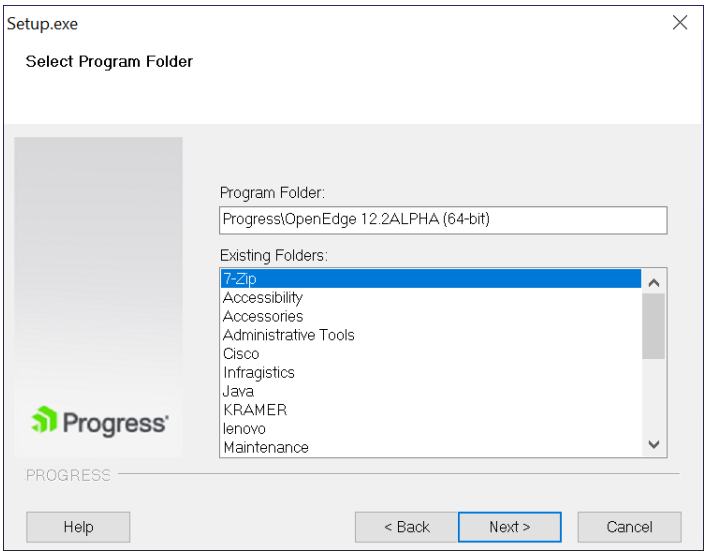
Note: Ensure that you have the correct JDK version installed. Otherwise, an error message is displayed, and you cannot proceed with the installation. For details about JDK requirements on each supported platform, see the *OpenEdge 12.2 Platform and Product Availability Guide*.

6. Specify the paths for the OpenEdge ESB adapter destination and working directories in the **Choose Destination And Working Path Directories** dialog box. Click **Next**.

   The working directory contains your OpenEdge database and application files, and is the directory where OpenEdge runs. The working directory cannot be same as the destination directory.



7. In the **Select Program Folder** dialog box, specify the Windows program folder where the OpenEdge ESB adapter program icons to appear. Click **Next**.



8. In the **Sonic ESB Properties** dialog box, enter the Sonic ESB properties and click **Next**.

   Enter the property values for Sonic ESB:

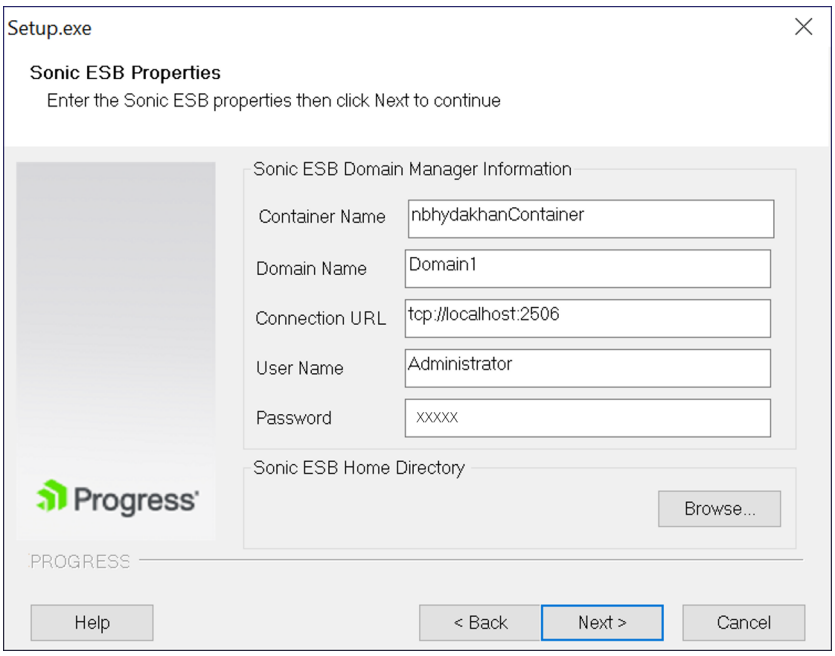
   a. Type a container name that is unique to the management broker, or accept the default host name that appears in the **Container Name** field.

   b. Accept the default value, `Domain1`, that appears in the **Domain Name** field, or type a new value that is associated with the value you entered for the container name.

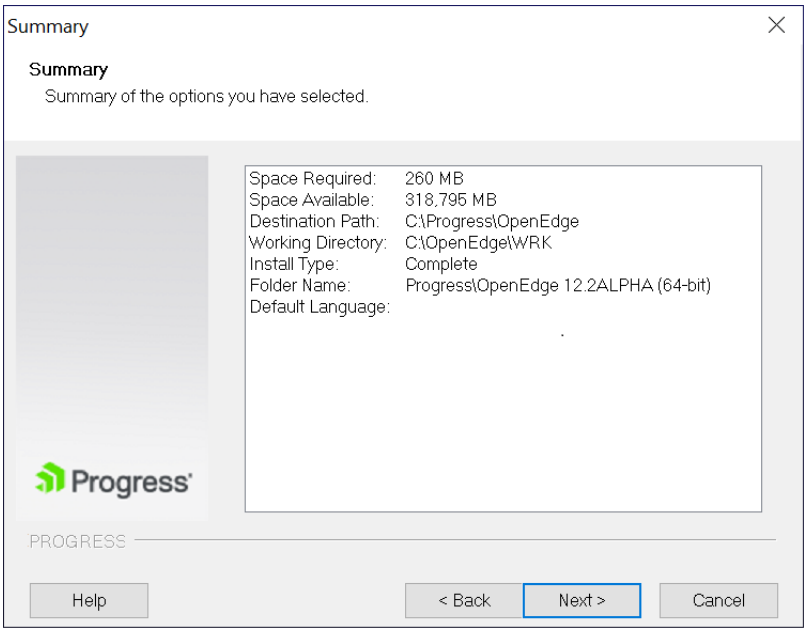
   c. Accept the default value, `tcp://localhost:`*`port`*, that appears in the **Connection URL** field, or enter a new value.

   d. Enter your administrator credentials.

   e. Specify the directory in which the Sonic ESB configuration files are to be stored.



9. In the **Summary** dialog box, review the installation options that you chose. Click **Back** to return to the previous dialog box to make any changes. If you are satisfied with your choices, click **Next**.



10. When the installation is complete, the **Typical Setup Done** dialog box is displayed. Click **Finish** to exit the installation process.

Install the OpenEdge ESB adapter on UNIX

The OpenEdge ESB installation program on UNIX or Linux machines presents a series of dialog boxes in which you enter your choices in each dialog box and advance at your own pace. The specific controls you use to advance to the next dialog box or return to a previous dialog box are identified on each dialog box.

Select a menu option using the spacebar, the TAB key, the cursor keys, or the accelerator keys that are highlighted in each selection on the dialog box.

Press CTRL+T to quit the installation at any time.

To get help on a dialog box, follow the instruction provided on the dialog box. Generally, you enter the control-key sequence or highlight the menu option, and then press ENTER. In the Help window, use the Up and Down arrow keys to scroll through the contents. Press ESC twice to return to the installation program.

To install the OpenEdge ESB adapter:

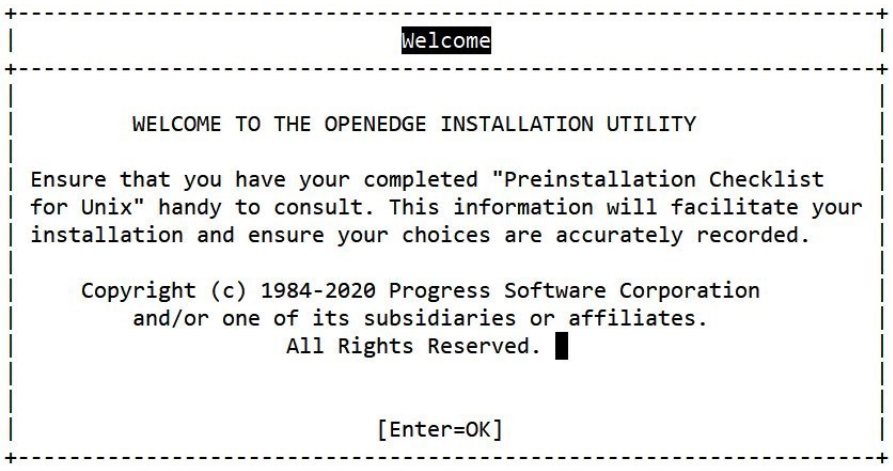
1. Log in to the machine as `root`.
2. If you have any applications running, then stop them.

   Running applications or tasks may interfere with the installation because they could use files that OpenEdge needs to complete the installation. Stop any processes that are running, and close any files that are located in the same directory where you want to install the OpenEdge ESB adapter.
3. Run the following command:

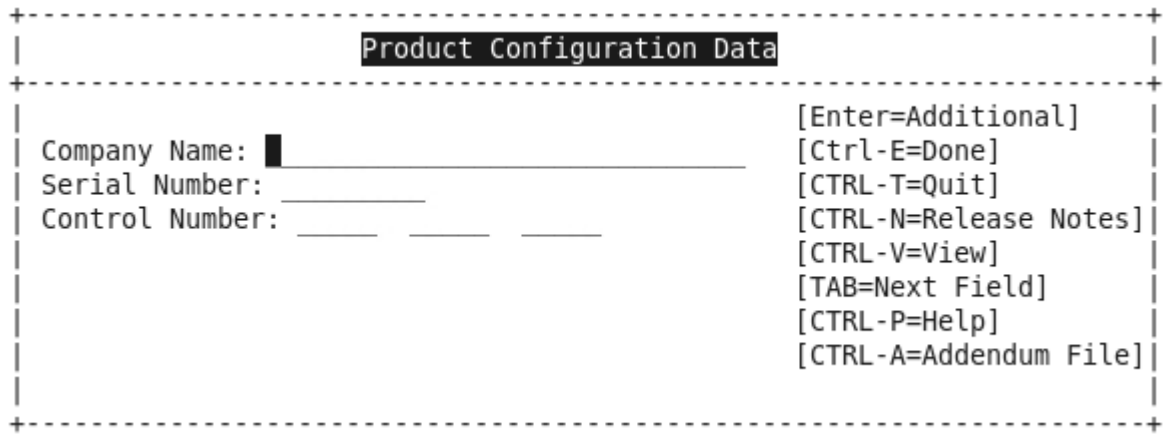
   ```
   ./path/proinst
   ```

   The **Welcome** dialog box appears.

   Note: In the command, `path` represents the directory to which you downloaded the ESB adapter software image. You must run the command from a directory other than `path` or the ESB adapter installation directory.
4. Read the information in the **Welcome** dialog box and press ENTER.



5. In the **Product Configuration Data** dialog box:
   a. (Optional) Type the company name in the **Company Name** field.
   b. Type the serial number in the **Serial Number** field.
   c. Type the control number in the **Control Number** field.



d. Press CTRL+E to submit your configuration information.

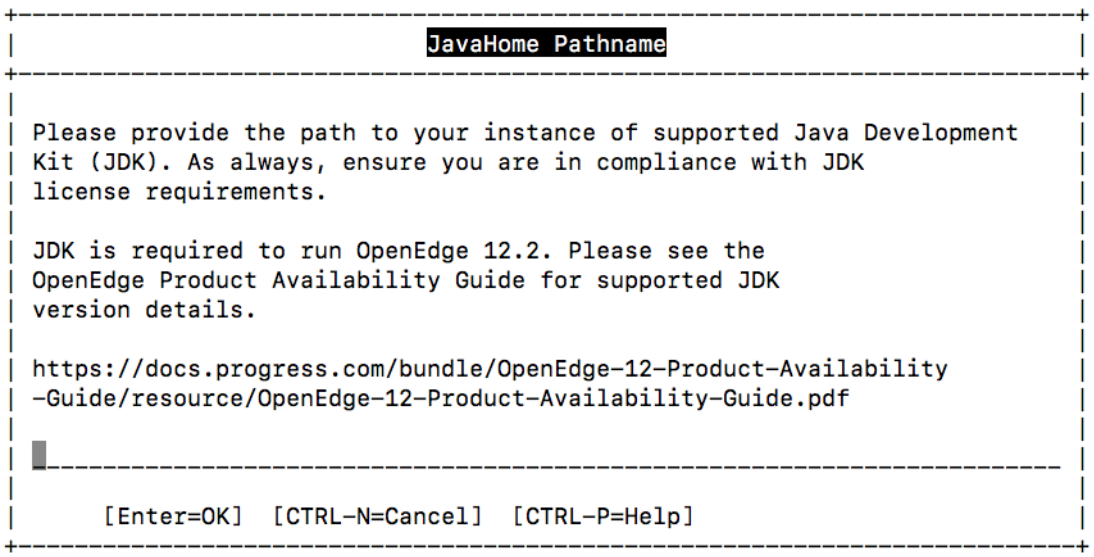
The **Done Configuration Data Confirmation** dialog box appears.

e. Type Y to confirm.

The **JavaHome Pathname** dialog box appears.

Note: Enter N to cancel and return to the **Product Configuration Data** dialog box.

6. Specify the JDK installation location in the **JavaHome Pathname** and press ENTER.



Note: Ensure that you have the correct JDK version installed. Otherwise, an error message is displayed, and you cannot proceed with the installation. For information about JDK requirements on each supported platform, see the OpenEdge 12.2 Platform and Product Availability Guide.

The **Install Type and Destination** dialog box appears.

7. In the dialog box:

   a. Specify the type of installation. The options are **Complete** or **Custom**. The default is **Complete**.

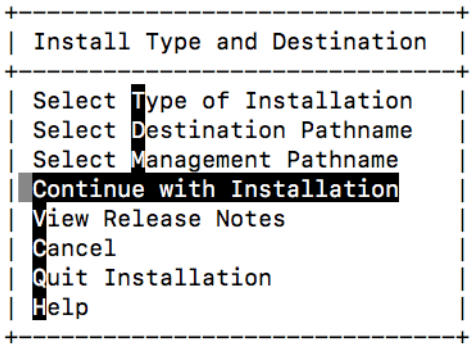
   Note: For custom installations, Progress recommends that you install all recommended products. If you do not install all the recommended products, then the adapter may not function as expected.

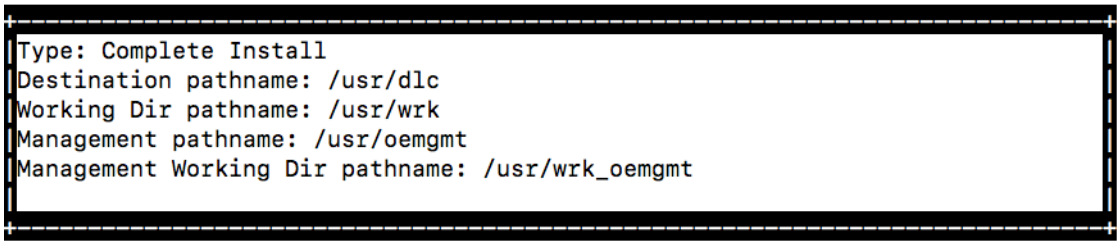
   b. Specify the installation directory. The default is `/usr/dlc`.

   c. Specify the working directory. The default is `/usr/wrk`.

   d. Specify the OpenEdge Management installation directory. The default is `/usr/oemgmt`.

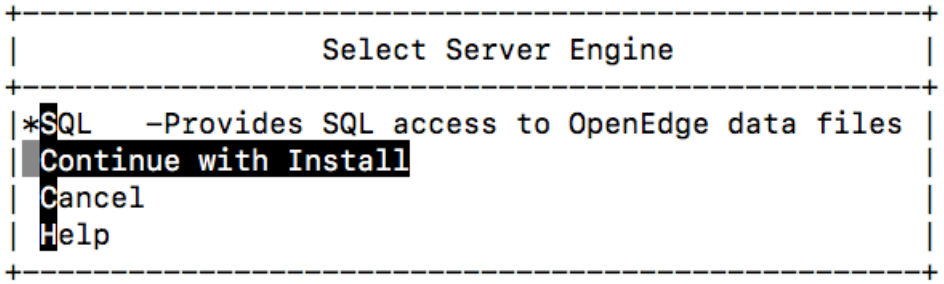
   e. Specify the OpenEdge Management working directory. The default is `/usr/wrk_oemgmt`.





   f. Press ENTER.

8. If the **Select Server Engine** dialog box appears, specify whether or not you want to install the SQL Server database engine and continue with the installation. By default, the database engine is installed.



9. In the **Sonic ESB Properties** dialog box:

   a. Type the container name in the **Container Name** field.

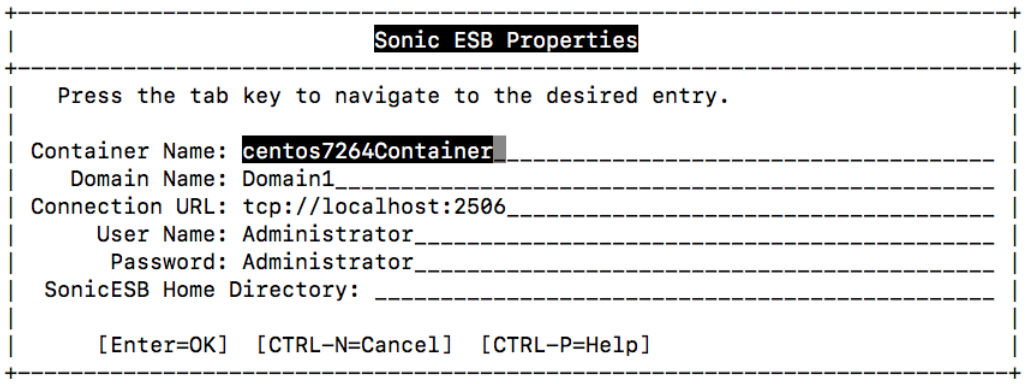
   b. Type the domain name in the **Domain Name** field.

   c. Type the connection URL in the **Connection URL** field.

   d. Type your user credentials in the **User Name** and **Password** fields.

   e. If you have Sonic ESB installed on the machine, then type the Sonic ESB installation directory in the **SonicESB Home Directory** field.

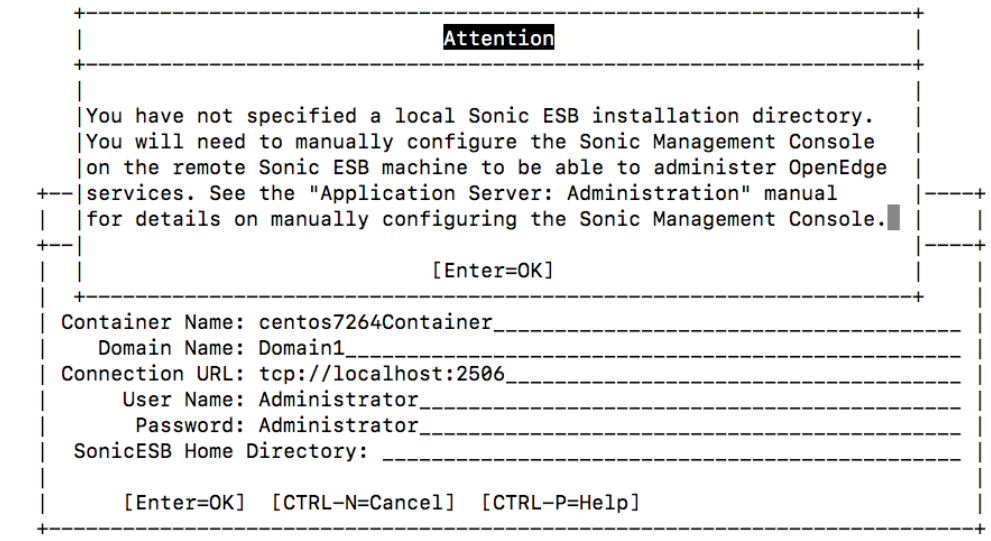
      If Sonic ESB is installed on a remote machine, then leave the **SonicESB Home Directory** field empty.



   f. Press ENTER.

      The **Complete Installation** dialog box appears.

Note: If the **SonicESB Home Directory** field is empty, then a message appears that states that you must manually configure the Sonic Management console on the remote machine.



---

10. In the **Complete Installation** dialog box:

    a. Review your installation choices.

    b. Type Y to confirm.

    If you want to modify your choices, then type N to return to the previous dialog box.

```
+------------------------------------------------------------+
|                    Complete Installation                   |
+------------------------------------------------------------+
|                                                            |
|The following products will be installed:                   |
|'OE Adap Sonic ESB'                                         |
|                                                            |
|Disk Space Required for Products: 223,068,160 bytes         |
|Disk Space Required for Installation: 223,907,840 bytes     |
|Disk Space Remaining After Installation: 92,169,854,976 bytes|
|                                                            |
|Selected Destination Path: /usr1/dlc                        |
|                                                            |
|Do you want to install the above listed product(s)?         |
|                                                            |
|                   [Y=YES] [N=NO] [H=Help]                  |
+------------------------------------------------------------+
```

11. In the **Copy Scripts** dialog box, confirm whether you want to copy the OpenEdge ESB adapter product scripts and program modules to the default location `/usr/bin`.

```
+-----------------------------+
|        Copy Scripts?        |
+-----------------------------+
|                             |
|Copy the scripts to /usr/bin?|
|                             |
|   [Y=YES] [N=NO] [H=Help]   |
+-----------------------------+
```

Note: If you are maintaining two versions of OpenEdge on the same machine, then type N to avoid overwriting files.

---

12. When the installation is complete, a message is displayed. Select **End the OpenEdge Installation**, and press ENTER to close the message box.

```
+------------------------------------------------------------+
|Installation of selected OpenEdge products is complete.     |
|Refer to the installation notes for more information.       |
+------------------------------------------------------------+
| End the OpenEdge Installation                              |
| View Release Notes                                         |
| Help                                                       |
+------------------------------------------------------------+
```

Install the OpenEdge ESB Adapter in silent mode

An interactive installation prompts you for information, and records the values you chose using a series of dialog boxes. When you start an interactive installation, the data that you enter is recorded in a response file. On UNIX and Linux machines, the default name for the response file is `linuxx86_64.ini`. On Windows machines, the default name for the response file is `response.ini`, which is created in the install subdirectory of your OpenEdge ESB adapter installation directory.

For more information about response files, see "Contents of the response file" in *Install OpenEdge*.

A silent installation performs an installation using a script, and is a two-step process:

1. Perform an interactive installation.
2. Execute the setup command and use the response file to provide values for the installation.

The installation data that is captured in a response file can be played back, to perform a silent installation through a batch mechanism. You can modify the contents of the response file that is generated by the installation program, or you can create a response file. However, for a silent installation, Progress recommends that you use the automatically generated and unmodified response file.

Note: A supported Java SE Development Kit (JDK) must be installed on your machine before running the OpenEdge Enterprise Service Bus (ESB) adapter silent installation program. The JDK is not included with the OpenEdge ESB adapter software. Ensure that you edit the location of the directory in which the JDK is installed in the response file before the installation procedure. For information about JDK requirements, see the *OpenEdge 12.2 Platform and Product Availability Guide*.

The command that you use to initiate or playback the response file is the same, regardless of which data input option you choose. The OpenEdge ESB adapter silent installer runs without intervention after you enter the command to start the process.

Syntax to install the OpenEdge ESB adapter on Windows

```
path-to-install-media\setup.exe -psc_s [-notify]
-psc_f1=path\response-file [-psc_f2=path\logfile]
```

In the preceding syntax:

- `path-to-install-media\setup.exe` runs the OpenEdge ESB adapter installation. You can run the installation from the directory that contains the installation executable that you downloaded from the Progress Download Center.
- `-psc_s` specifies a silent installation.
- `-notify` specifies that the installation dialog boxes that are displayed contain details about the current installation phase and the completion percentage.

  Progress recommends that you set up your installation to poll the log file for the status of the installation. See "Check the status of the silent installation log file" in *Install OpenEdge*, to review the contents of the Runtime Status and Result Code section in the sample `oesetup.log` file.
- `-psc_f1=path\response-file` specifies the full path of the response file. By default, the silent installation searches for the response file `response.ini` in the same directory that `setup.exe` is located.
- `-psc_f2=path\logfile` creates an installation log file. If no path or name is specified, then the file `oesetup.log` is created in the Windows directory by default.

Example

The following command executes a silent installation on Windows:

```
\\cd-server\OpenEdge\setup.exe
-psc_s -psc_f1=C:\SilentInstalls\response.ini
    -psc_f2=C:\SilentInstalls\oesetup.log
```

Syntax to install the OpenEdge ESB adapter on UNIX

```
proinst -b path/install-ini-name -l
path/logfile [-n]
```

In the preceding syntax:

- `proinst` is the command that initiates the OpenEdge ESB adapter installation.
- `-b` *`path/install-ini-name`* performs a batch installation, specifying the full path of the `.ini` file to be used for the silent installation. You can use the `linuxx86_64.ini` file or another `.ini` file that you create.
- `-l` *`path/logfile`* creates a log file, optionally specifying the full path of the log file in which the installation events are logged. By default, the log file `install.log` is created in the directory identified by the first environment variable that the installer finds among the following: `$TMP`, `$TEMP`, or `$TMPDIR`.
- `-n` specifies that the batch installation will include a progress meter that displays details about the current installation phase and completion status.

Example

The following command executes a silent installation on a UNIX:

```
proinst -b /test/linuxx86_64.ini -l /log/test.log
```

3

Guidelines for Using and Programming for the OpenEdge Adapter for Sonic ESB

This topic discusses general considerations related to OpenEdge Sonic ESB services and the operation of the OpenEdge Adapter for Sonic ESB. You can also find more detailed information in other documentation, as indicated in this topic.

For details, see the following topics:

- Native Invocation methodology
- Configure Progress Developer Studio for OpenEdge for ESB annotations
- Web Services Invocation methodology
- Service definition considerations
- Sonic message handling run-time parameters

Native Invocation methodology

The **OpenEdge Adapter for Sonic ESB** supports a Native Invocation methodology for exposing ABL applications to the **Sonic Enterprise Service Bus (ESB)**. When a Sonic ESB process is created using the Native Invocation methodology, ABL procedures are called directly through an OpenAPI call to PAS for OpenEdge.

The Native Invocation methodology provides the following benefits:

- Simplified exposure of ABL code as a service

- Simplified process of mapping ABL parameters to Sonic messages
- Reduced overhead improving run-time performance over the Web Services Invocation methodology by eliminating conversions to and from SOAP messages

The Native Invocation methodology relies on invocation (`.esboe`) files that are imported directly into Sonic Workbench.

Main steps to expose an ABL procedure:

1. Develop your ABL code and create invocation files.
2. Import your invocation (`.esboe`) files into your Sonic project in Sonic Workbench.
3. Create a **Sonic ESB Process**, adding your invocation files to the process, one at a time, and mapping your input and output parameters to message definitions.
4. Save your **ESB Process** and upload it to your Sonic Domain for testing and deployment.

This process can be further simplified if you have Sonic V7.6 or later and have integrated Progress Developer Studio for OpenEdge and Sonic Workbench into one Eclipse environment.

The topics that follow discuss these development steps in greater detail. For a complete example of creating and testing an ESB process using a native invocation file, see Run a sample Sonic ESB Process using Native Invocation on page 59.

Create an invocation file

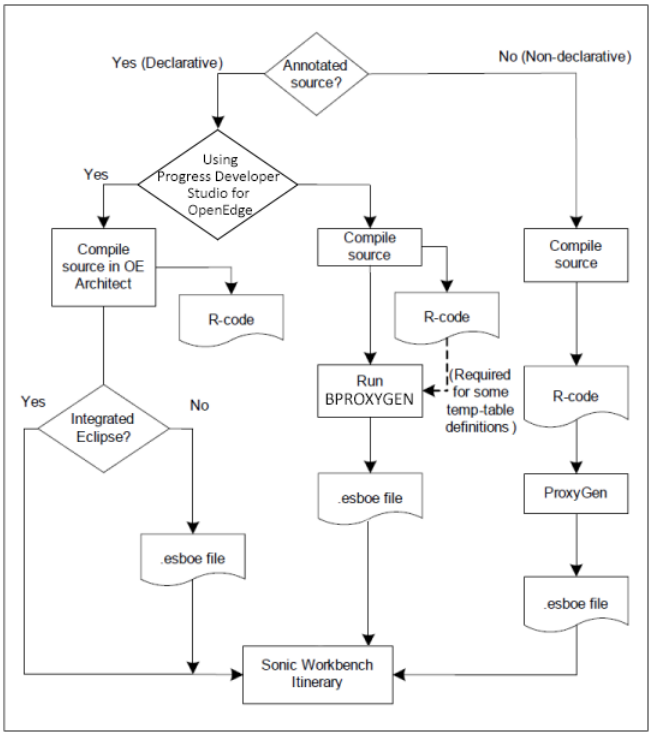
OpenEdge developers have the choice of two approaches for creating an invocation file for ABL procedures and functions:

- **Declarative**—Capture information about publicly exposable procedures in the source code through the use of annotations.
- **Non-declarative**—Capture information about publicly exposable procedures through the use of a utility such as ProxyGen.

The declarative approach is recommended to OpenEdge developers as a best practice. You enter relevant information for making a procedure, function, or external procedure publicly exposable when the source code is written. This information is then stored with the source code, and during the build process is captured as part of the r-code. If you do not want to capture information about publicly exposable procedures with the source code, then use the non-declarative approach.

In OpenEdge, at development time, the ability to drag and drop ABL procedure files directly into a Sonic ESB itinerary is supported. Dragging ABL procedure files directly into a Sonic ESB itinerary eliminates the need to create and import an `.esboe` file. If your source code is not already annotated with ESB annotations when you do the drag and drop, then a wizard walks you through the steps required to add the necessary native invocation annotations.

The creation of an ESB itinerary based on `.esboe` files are fully supported. The following figure graphically depicts the creation of an invocation `.esboe` file, using the declarative or non-declarative approach.



Declarative invocation files

Declarative invocation files rely on annotations. **Annotations** capture information in source code that extends the ABL language syntax. ABL source code annotations signal that external procedures, internal procedures, and user-defined functions are available to Sonic ESB processes. For external procedures, the annotation must precede all ABL statements in the source file. For internal procedures and user-defined functions, the annotation must precede the `PROCEDURE` or `FUNCTION` statement.

Annotation syntax

The native invocation annotation syntax is the constant `"@openapi.openedge.export"` followed by the scoping value `"FILE"` for external procedures, and then followed by a series of name-value pairs. The following table describes the annotation name-value pairs.

Name	Default value	Mandatory/ Optional	Description
`type`	ESB	Mandatory	Specifies the type of open client the procedure or function is exported to. The only type currently supported is ESB.

Name	Default value	Mandatory/ Optional	Description
`esboeFileName`	For external procedures: %FILENAME% For internal procedures and functions: %FILENAME%_ %PROCNAME%	Optional	Specifies the name of the `.esboe` file. The extension is automatically appended. There is one file per annotation. If *esboeFileName* is not specified, then the default value is used.
`executionMode`	external	Mandatory	Specifies that the procedure runs persistently or externally. Specify `executionMode ="persistent"` for a file's external procedure before internal procedures and functions can be annotated. `executionMode="persistent"` can only be specified for top-level external procedures.
`useReturnValue`	false	Optional	Specifies whether a return string is generated for a procedure. Specify `useReturnValue="true"` to generate a return string. If `useReturnValue` is not specified, then no return string is generated.
`writeDataSetBeforeImage`	false	Optional	Specifies whether before-image data is written when `ProDataSet` parameters are serialized as XML. Specify `writeDataSetBeforeImage="true"` to write the before-image data. Specify `writeDataSetBeforeImage="false"` to not write the before-image data. If no `ProDataSet` parameters exist, `writeDataSetBeforeImage` is ignored.

Annotation examples

You can automatically add annotations in Progress Developer Studio for OpenEdge, or you can add them manually. For more information about adding annotations in Progress Developer Studio for OpenEdge, see Annotate ABL in Progress Developer Studio for OpenEdge on page 38. The following code illustrates annotations:

- External procedure

The following code illustrates the annotation of an external procedure, `foo.p`. The only required name-value pairs are `type` and `executionMode`, as shown:

```
/* foo.p */
@openapi.openedge.export FILE (type="ESB",executionMode="external").
DEFINE INPUT PARAM bar AS INT.
DEFINE OUTPUT PARAM ney AS CHAR.
...
```

Processing this annotation creates the file `foo.esboe` that describes a non-persistent operation named `foo`. `foo` takes an input integer named `bar` and returns a character string named `ney`.

- Renamed `.esboe` file for an external procedure

  The following code illustrates the annotation of an external procedure, `foo.p`, that renames the generated `.esboe` file:

```
/* foo.p */
@openapi.openedge.export FILE (type="ESB", ,executionMode="external",
         esboeFileName="Renamed_foo").
DEFINE INPUT PARAM bar AS INT.
DEFINE OUTPUT PARAM ney AS CHAR.
...
```

This annotation creates the file `Renamed_foo.esboe` that describes a non-persistent operation on the PAS for OpenEdge.

- Internal procedure

  The following code illustrates the annotation of an internal procedure, `barr`, inside the file `foo.p`:

```
/* foo.p */
@openapi.openedge.export FILE (type="ESB", executionMode="persistent").
DEFINE INPUT PARAM bar AS INT.
DEFINE OUTPUT PARAM ney AS CHAR.
...@openapi.openedge.export(type="ESB")
PROCEDURE barr.
DEFINE INPUT PARAM abc AS INT.
DEFINE OUTPUT PARAM xyz AS CHAR.
...
```

The external procedure `foo` must be annotated as persistent before the internal procedure `barr` can be annotated. Three files are created based on these annotations:

- **`foo.esboe`**—Runs the external procedure persistently
- **`foo_barr.esboe`**—Invokes the internal procedure
- **`foo_release.esboe`**—Deletes the persistent procedure, unbinding PAS for OpenEdge from the process

Annotate ABL in Progress Developer Studio for OpenEdge

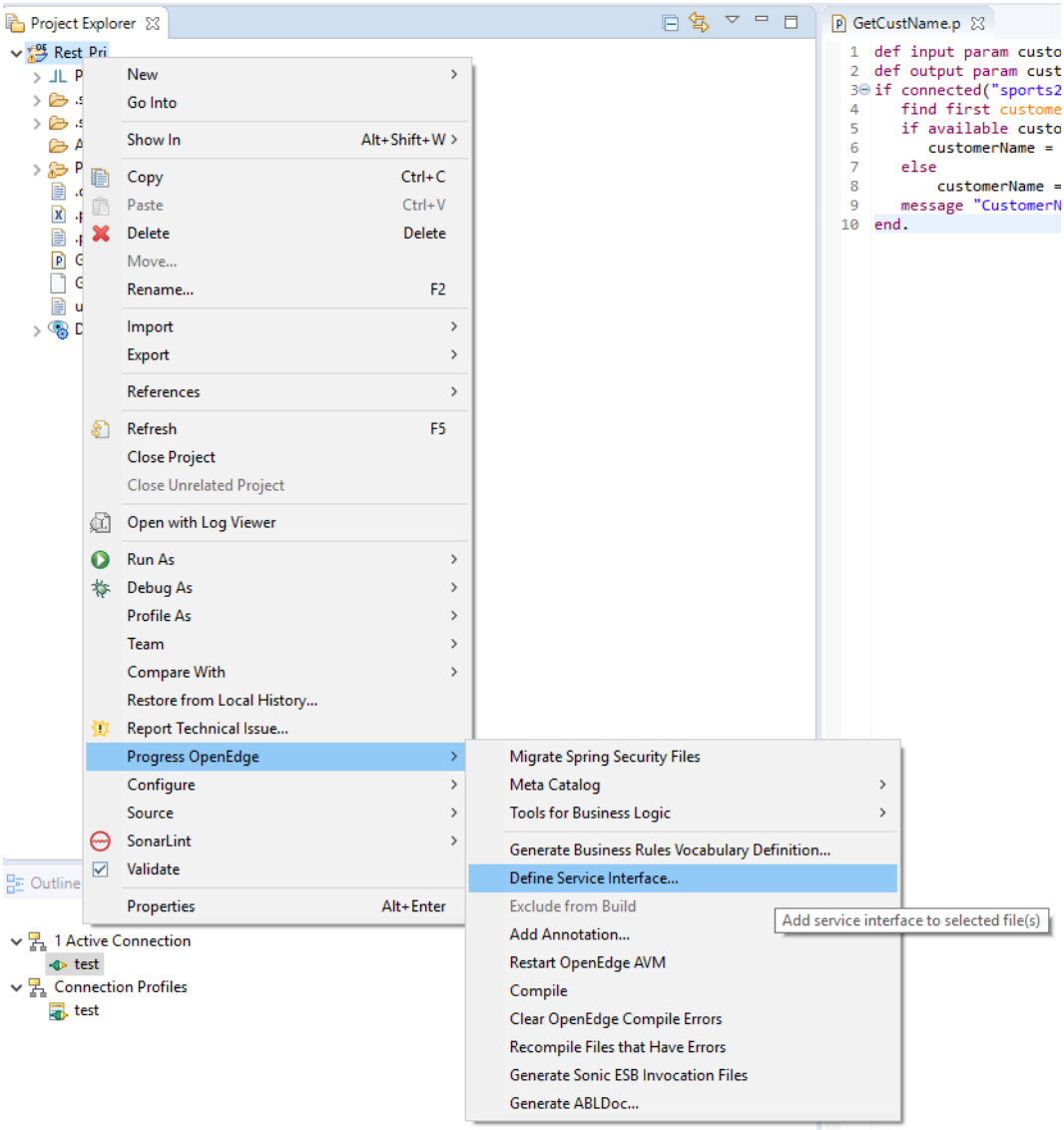
Progress Developer Studio for OpenEdge's annotation capabilities include the creation of ESB native invocation annotations and the generation of `.esboe` files. See Configure Progress Developer Studio for OpenEdge for ESB annotations on page 51 to verify that Progress Developer Studio for OpenEdge is configured to generate native invocation annotations. After you configure your project, you are ready to annotate your ABL source code.

For more information about different methods on "Add Annotations", see *Progress Developer Studio for OpenEdge Online Help*.

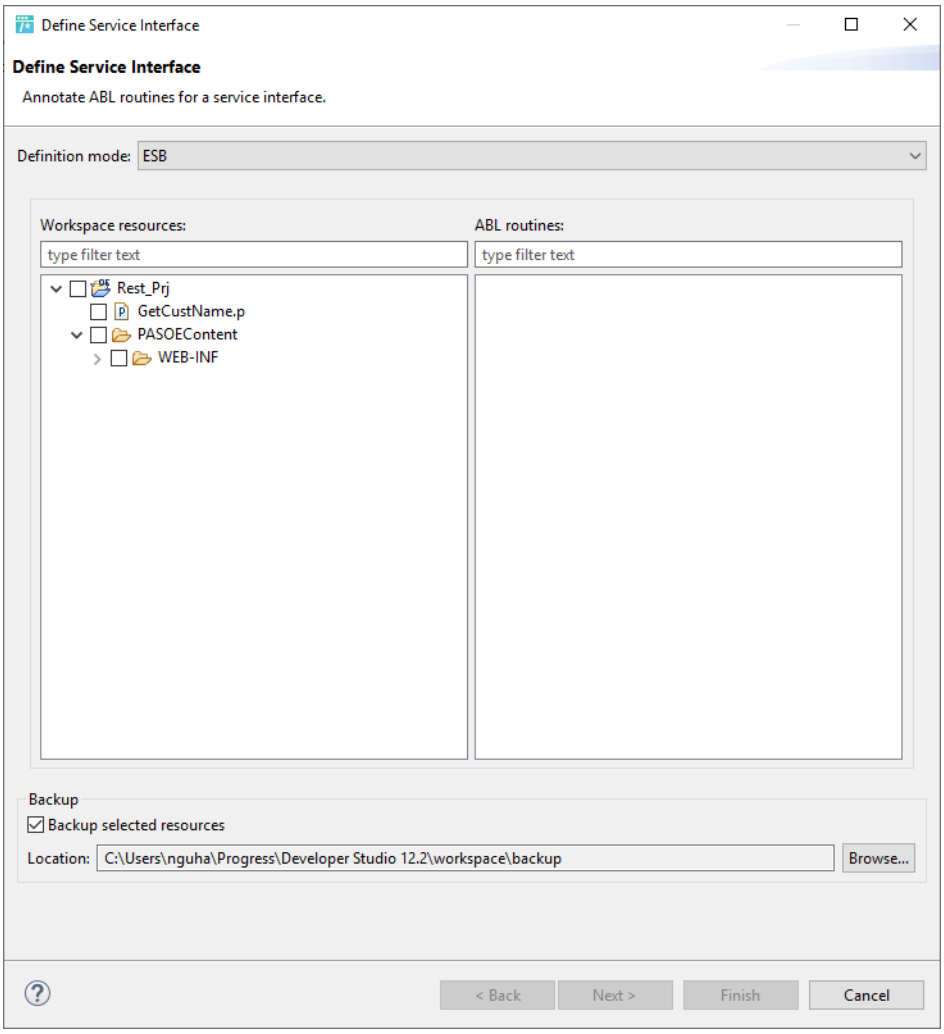
Annotate multiple source files at once

To annotate multiple source files at once:

1. Start the wizard by right-clicking the project and selecting **Progress OpenEdge** > **Define Service Interface**.



The **Define Service Interface** dialog box is displayed:



2. Select **ESB** from the **Definition mode** drop-down list.
3. Select the files to annotate in **Available Resources**.
4. Click **Finish** if you are only annotating external procedures, or click **Next** to also annotate persistent procedures and functions.
5. Select the procedures and functions you want annotated and click **Finish**.

Generate .esboe files

After you annotate your source, you generate your `.esboe` files. Progress Developer Studio for OpenEdge provides two options for generating the files:

- If you configured your preferences to automatically build the `.esboe` files, build your project. For instructions on configuring your project, see Configure Progress Developer Studio for OpenEdge for ESB annotations on page 51.
- Right-click on either the **Resources** view or the ABL Editor, and select **OpenEdge** > **Generate Sonic ESB Invocation Files**.

Outside of Progress Developer Studio for OpenEdge, use ProxyGen to generate your `.esboe` files from annotated ABL code. For more information about ProxyGen, see *Use the Open Client Toolkit*.

Note: You can generate `.esboe` files from unnannotated ABL code with ProxyGen. For more information, see Non-declarative invocation files on page 40.

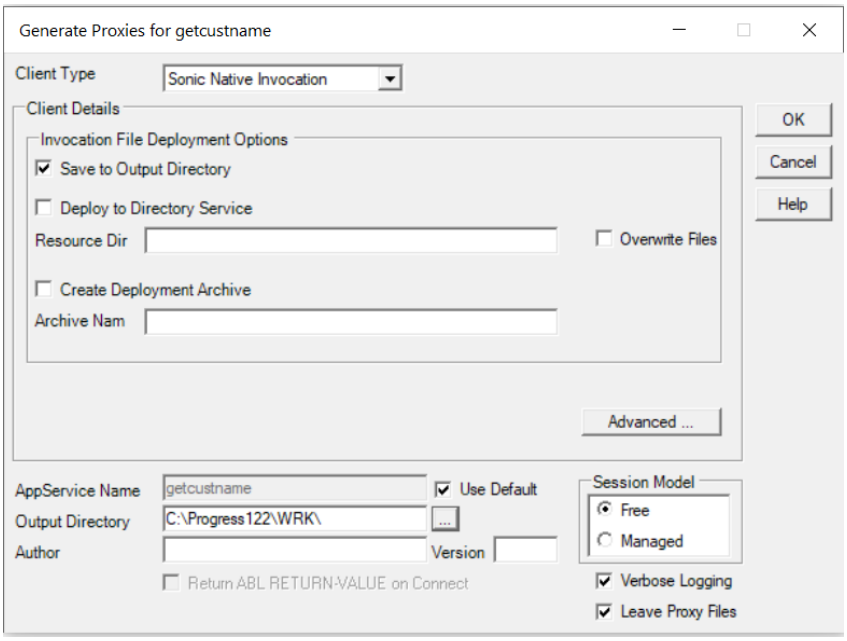
After you generate your `.esboe` files, you are ready to create an ESB process in Sonic.

Non-declarative invocation files

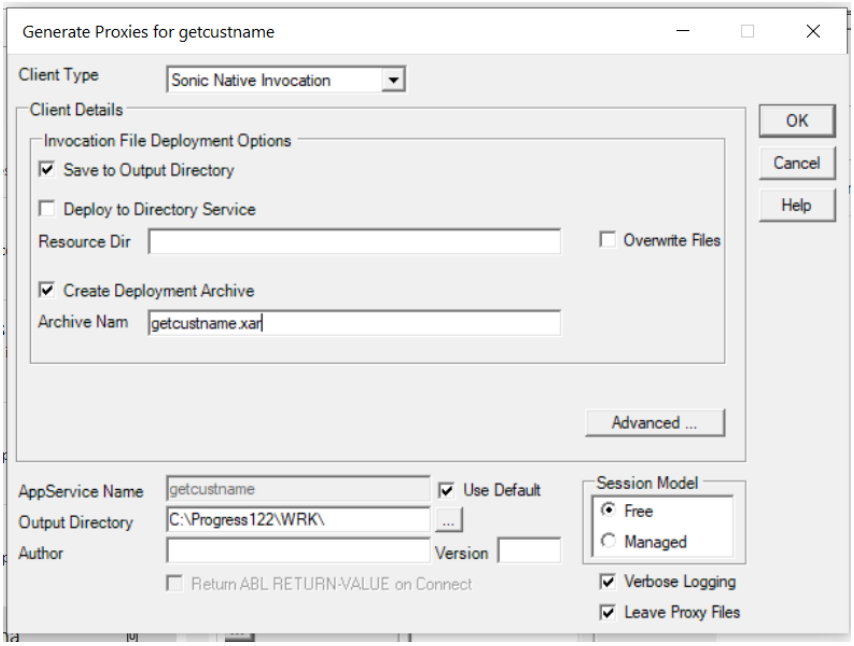
Non-declarative invocation files are generated from unannotated ABL code using ProxyGen.

To generate `.esboe` files with ProxyGen:

1. Ensure that your ABL procedures are compiled.
2. Start ProxyGen and create a new **AppObject**. Provide appropriate components and select procedures.
3. Select **File** > **Generate** to bring up the **General** tab of the **Generate Proxies** dialog box, as shown:



4. Select **Sonic Native Invocation**, and specify the destination directory for your `.esboe` files in the **Output Directory** field.
5. Select the **Sonic Native Invocation** tab, as shown:



6. Select the output destination of your `.esboe` files, and click **OK** to generate them. If you check **Save to Output Directory**, then your `.esboe` files are saved to the directory you specified on the **General** tab. If you select **Deploy to Directory Service**, then you must specify the absolute path of your Sonic Directory Service in the **Resource Dir** field. If you select **Create Deployment Archive**, then you must specify an archive (`.xar`) name in the **Archive Name** field.

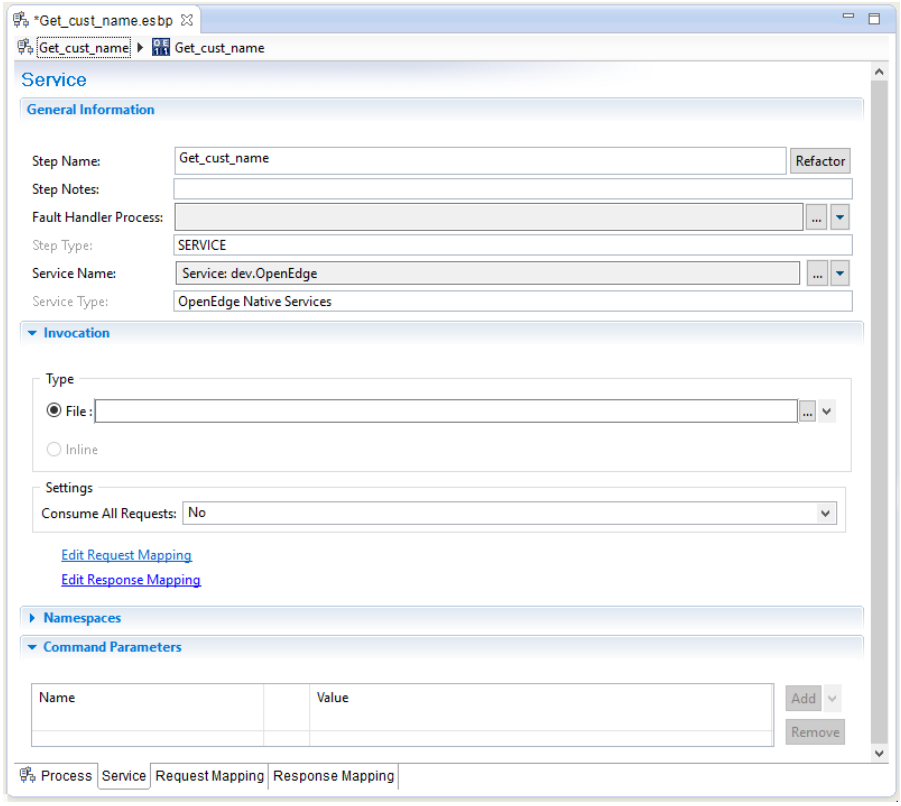
Import Native Invocation files into Sonic Workbench

After you develop your ABL procedures and created your native invocation files, proceed to **Sonic Workbench** to import your invocation (`.esboe`) files and develop your ESB process.

Note: The following steps are not intended as a detailed tutorial for Sonic Workbench. For details about Sonic Workbench, see your Sonic documentation.

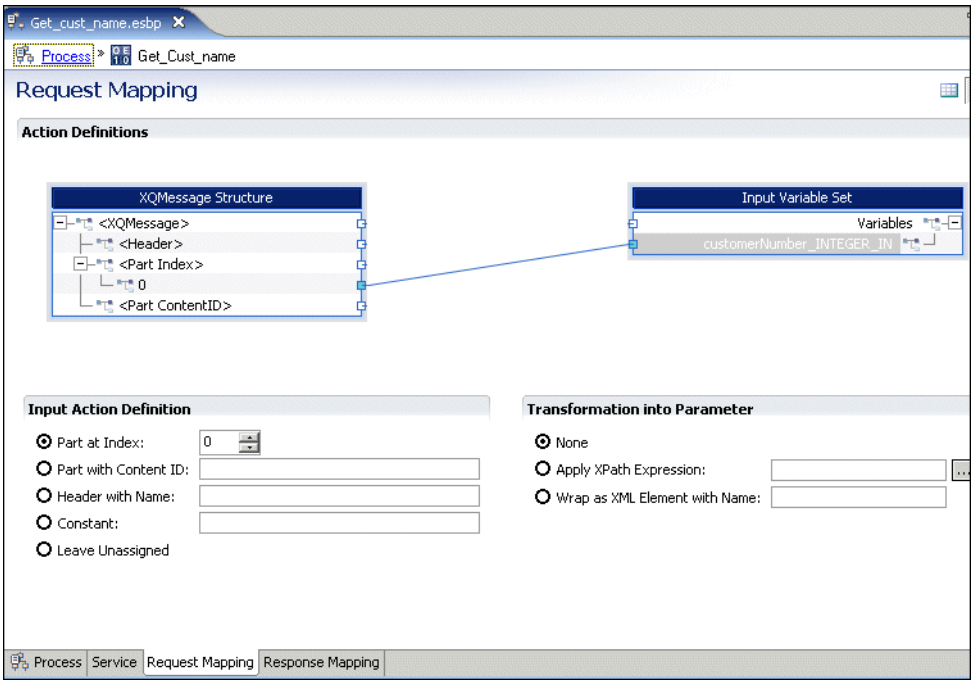
To import your native invocation files into an ESB process:

1. Start Sonic Workbench, and open an existing project or create a new one.
2. Choose **File** > **Import** to import your invocation files into the Sonic Directory Service.
3. On the process design canvas, select **OpenEdge Native Services** from the **Palette**, and drop it on the canvas. Give the service a meaningful name.
4. Provide the service information. You can drag an invocation file from the **Navigator** tab onto the service, or right-click on the service and choose **Open**.



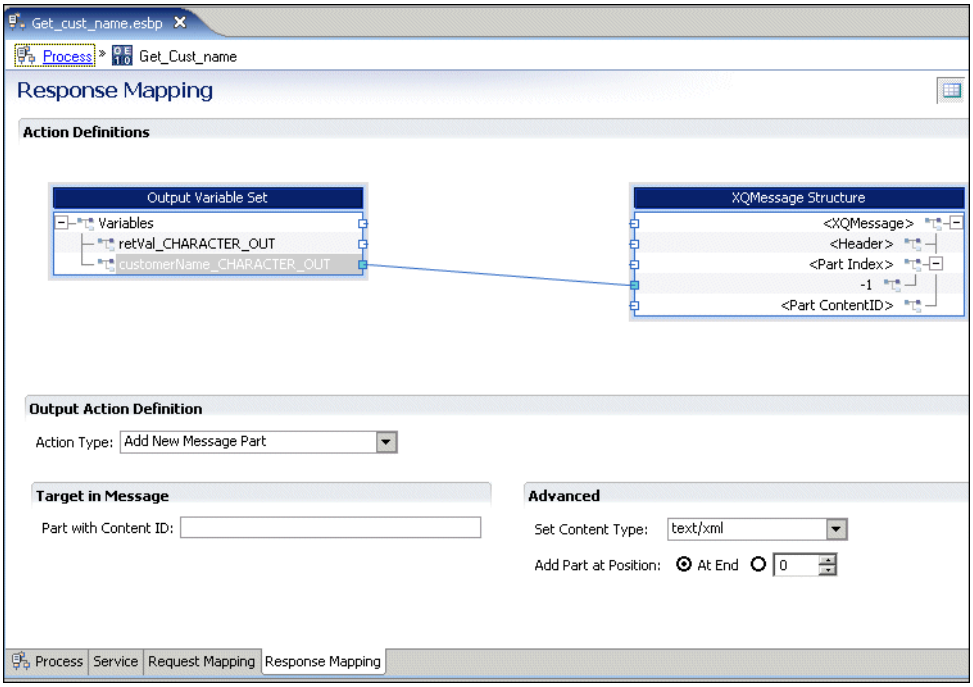
If you dragged an invocation file onto the service, then **Invocation File** displays the filename. If you did not drag a file, then browse to the appropriate invocation file.

5. Click the **Request Mapping** tab:



The initial input message parts are mapped to the input parameters of the ABL procedure. Each parameter can be mapped to specific message parts or message headers, set to constants or ignored. The data can also be transformed using XPath or wrapped in an XML element.

6. Click the **Response Mapping** tab:



The output parameters can be placed in message parts or message headers. The outpur parameters can be inserted as XML elements or attributes, or they can be discarded.

7. Complete the development of your process: add additional invocation files or other services.

ABL parameter naming

The **Request Mapping** and **Response Mapping** tabs show the list of input and output parameters and the XML schema types that are required to invoke the operation. Each parameter can be mapped to the specific message parts or to messages headers, set to constants, or ignored. For request parameters, the data can also be transformed using XPath or wrapped in an XML element. For response parameters, the data can be placed in message parts or headers, inserted as XML elements or attributes, or discarded.

To help distinguish the ABL datatype of parameters, a special naming convention is used. Each parameter is identified by its ABL parameter name, followed by its ABL type, and the mode of the parameter (`IN` or `OUT`). The construction is:

```
ABLparameter_ABLdatatype_ABLparameter-type
```

The following table shows the resulting ESBOE parameters names if you expose an ABL procedure that has an integer named `CustNum` as the input parameter, and returns a string named `CustName`.

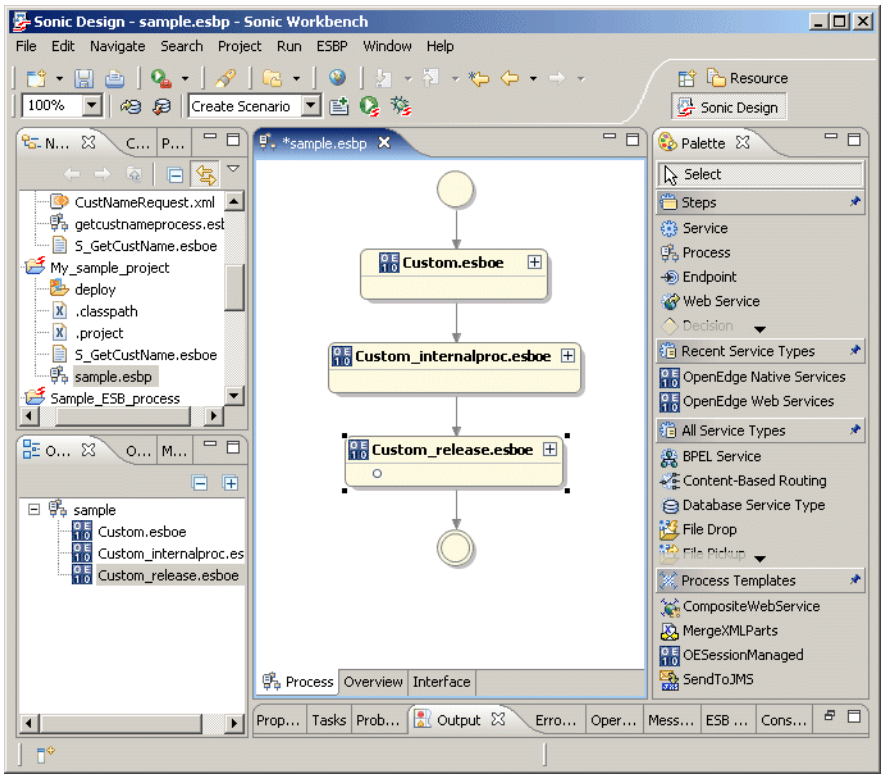
ABL parameter	ABL datatype	ABL parameter type (IN/OUT)	ESBOE parameter
`CustNum`	`INTEGER`	IN	CustNum_INTEGER_IN
`CustName`	`CHARACTER`	OUT	CustName_CHARACTER_OUT

Internal procedure invocation

Generating a native invocation file for an internal procedure or user-defined function, creates a total of three invocation files: one to run the external procedure persistently, one to invoke the internal procedure, and one to delete the persistent procedure and unbind PAS for OpenEdge from the process. All three invocation files must be imported into your ESB process in order.

The following figure shows the three invocation files in a sample process.

Figure 1: Internal procedure invocation



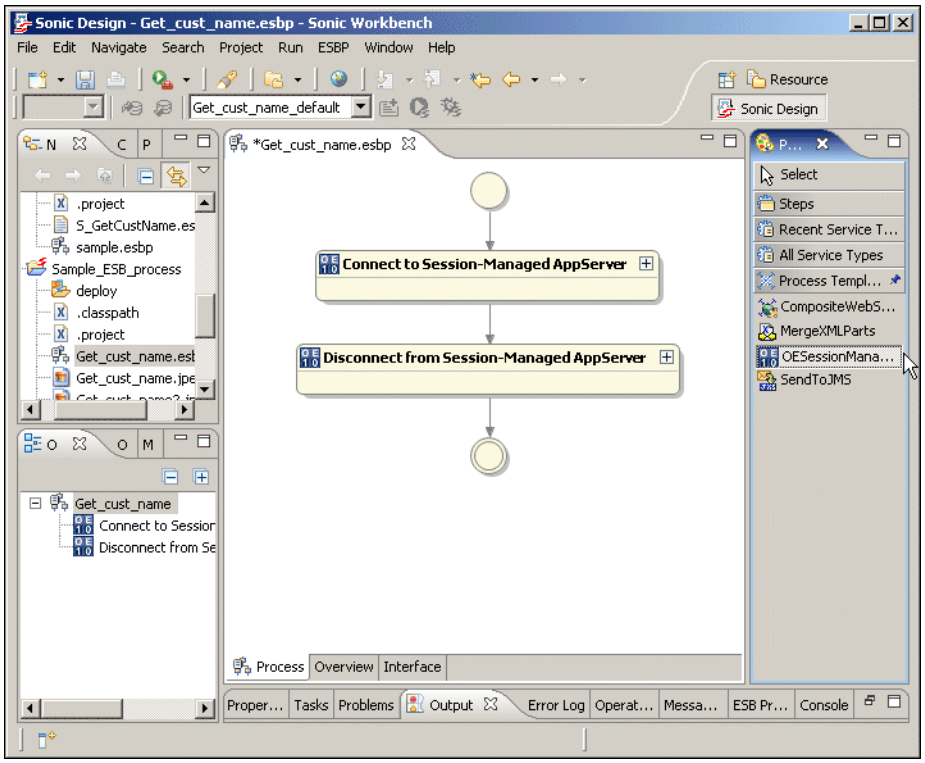
Calling the external procedure creates a `proc-id` parameter that is mapped throughout the process.

ESB process and session-managed PAS for OpenEdge

Your `.esboe` files do not contain information about your PAS for OpenEdge operating mode. OpenEdge installs process templates into your Sonic Directory Service for you to include in your process if you need a session-managed connection. The process template, **OESessionManaged**, adds two invocation files to your process: **Connect to Session-Managed AppServer** and **Disconnect from Session-Managed AppServer**. By default, the connection is made to the default `dev.OpenEdge` service in the `dev_OpenEdgeTest` container. You can change the service properties of your `dev.OpenEdge` service in your Sonic Management Console, or you can create a new service for your session-managed PAS for OpenEdge instance. If you create a new service, the you must reconfigure all your invocation files in your process to connect to your new service in your Sonic Workbench process editor.

Best practices for process design dictates that a session-managed session should not span more than a single process. Change your process design to use subprocesses if you require multiple session-managed PAS for OpenEdge.

The following figure shows the OESessionManaged template in Sonic Workbench.



Drag and drop ABL files

Dragging ABL procedure files directly into a Sonic ESB itinerary eliminates the need to create and import an `.esboe` file. Successfully dragging and dropping files requires that your environment be configured as follows:

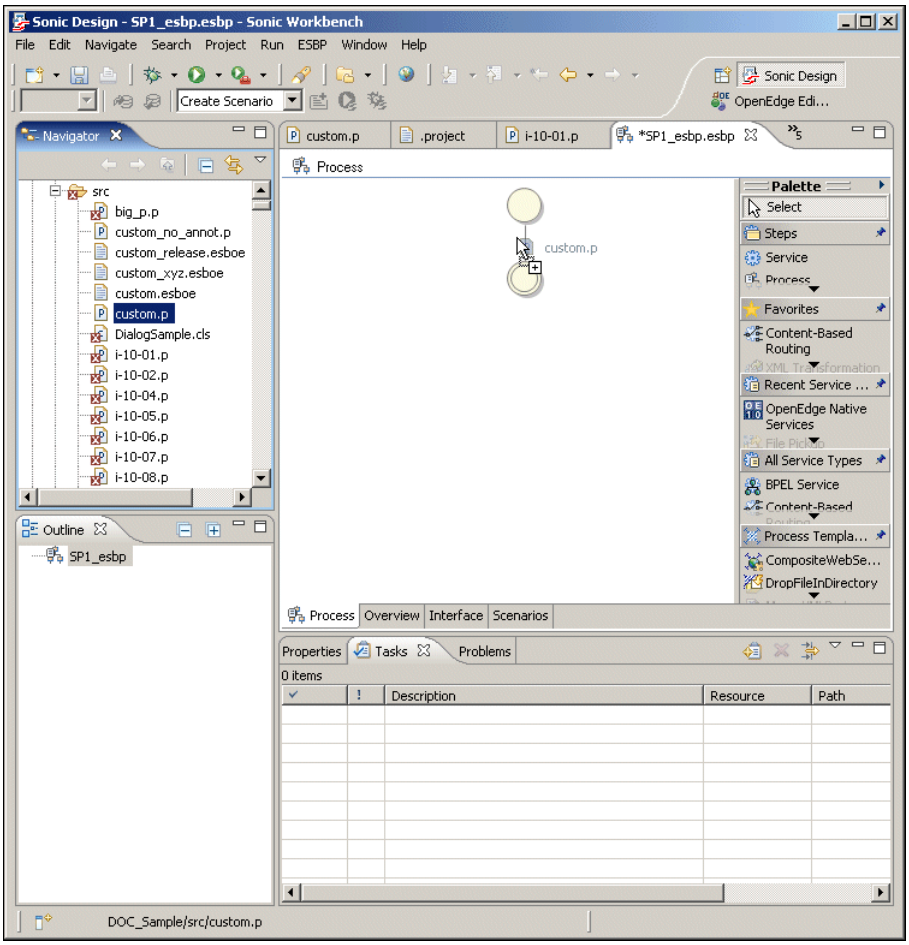
- Software
  - OpenEdge Release 10.1C01 (Service pack 1) or later for Windows. The Progress Developer Studio for OpenEdge and the OpenEdge Adapter for Sonic ESB products must be installed.
  - Sonic Workbench 7.6 or later.
- Configuration — Your Progress Developer Studio for OpenEdge and Sonic Workbench Eclipse environments must be integrated. See Progress Software Developers Network (PSDN) for a matrix of supported versions and detailed instructions on integrating the two environments.

Drag and drop ABL files into an ESB process

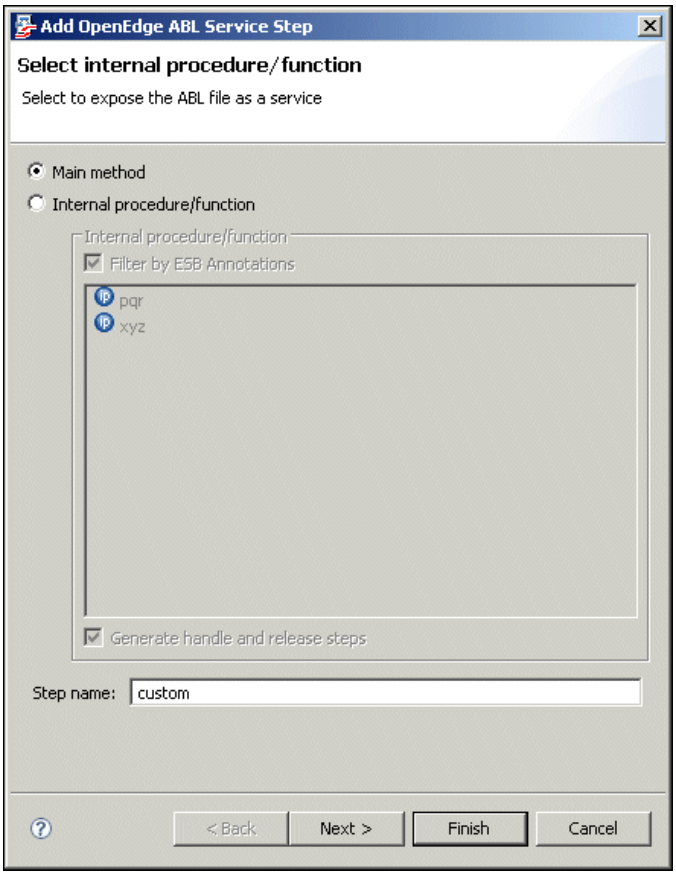
After your environment is configured, dragging and dropping your ABL procedure files into an ESB process itinerary is as simple as selecting the file and placing in the process editor.

To drag and drop an ABL file into an ESB process:

1. Open an ESB process editor window. Select an ABL file from an OpenEdge project in the **Navigator** window, and drag it into the ESB process:



2. After dropping your ABL procedure file, a wizard guides you through the process of selecting the procedure you are including in your ESB process. Complet the wizard as follows:



a) Sselect either **Main method** or **Internal procedure/function**.

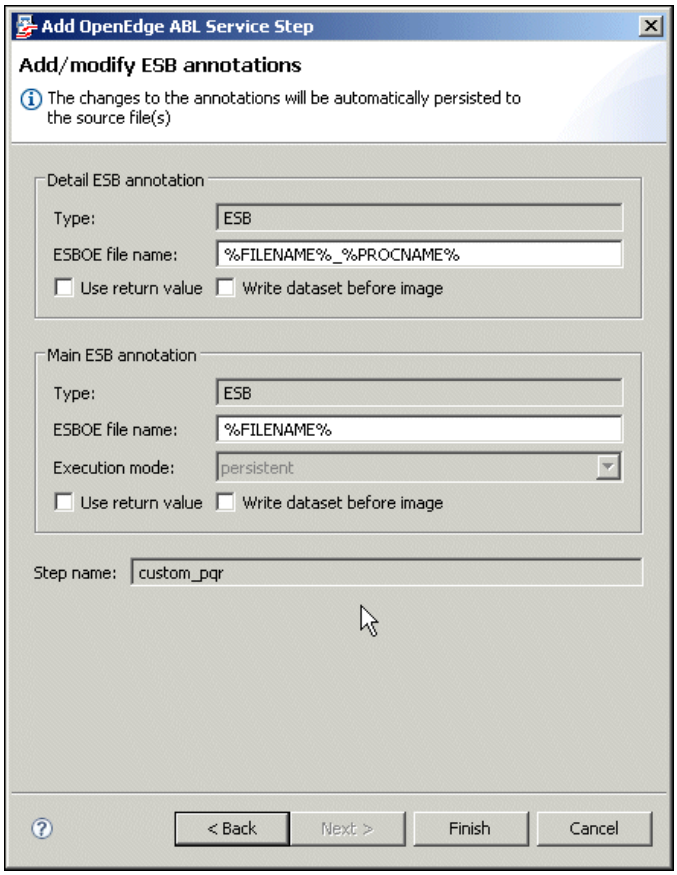
   If you select **Internal procedure/function**, you can select **Filter by ESB Annotations** to sort annotated procedures and functions higher in the selection box.

b) Check **Generate handle and release steps** to run the external procedure persistently before calling the internal procedure, and to delete the persistent procedure after the internal procedure is run.

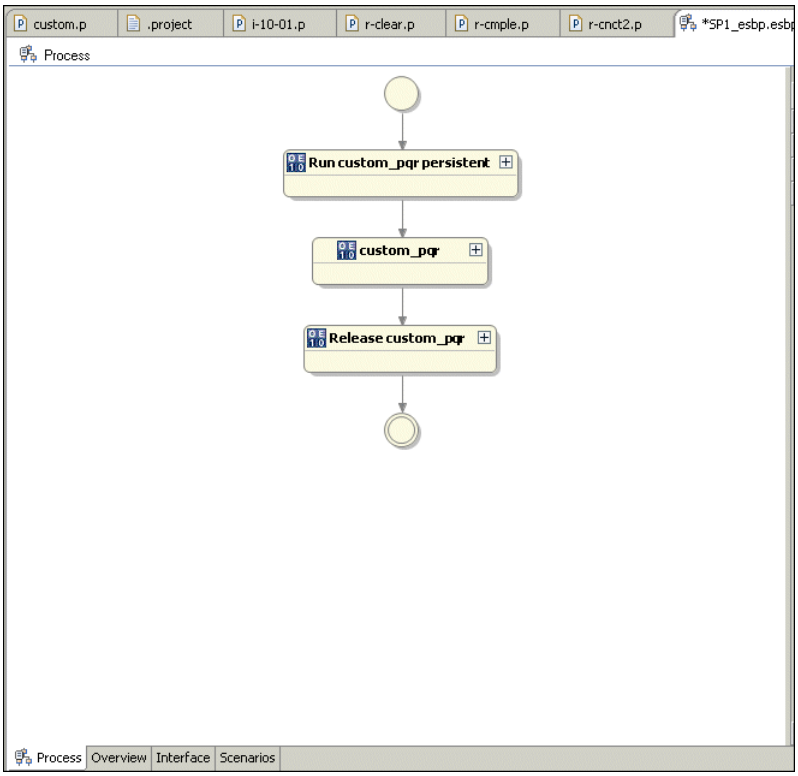
   If you are adding steps that call more than one internal procedure from the same ABL file, you only need one set of handle and release steps.

c) Create a **Step Name**. By default, **Step Name** defaults to the ABL file name (without the `.p` suffix) for the main procedure, and the ABL file name and the internal procedure name separated by an underscore for an internal procedure.

d) If your ABL source contains native invocation annotations, click **Finish**. If your ABL source is not annotated, or you need to modify the annotations, click **Next**. The following dialog box opens:



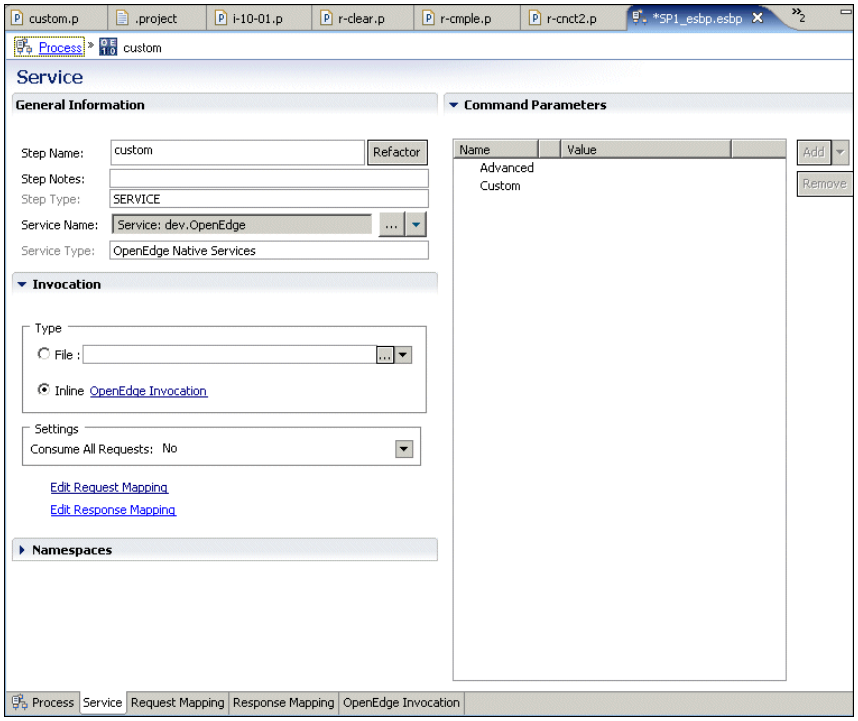
3. If you have selected an internal procedure, then both the **Detail ESB annotation** and **Main ESB annotation** sections appear. For an external procedure, only the **Main ESB Annothation** section appears. For details about completing the annotation information, see Annotate ABL in Progress Developer Studio for OpenEdge on page 38 for a complete definition of the native invocation annotation components. Click **Finish**.
4. After you complete the annotation wizard, your ABL procedure is added to the ESB process. An internal procedure added to the ESB process is shown:



If you modify the signature of your ABL procedure, such as the number of parameters, or parameter datatypes, then you must delete and re-add the ESB process step for the procedure.

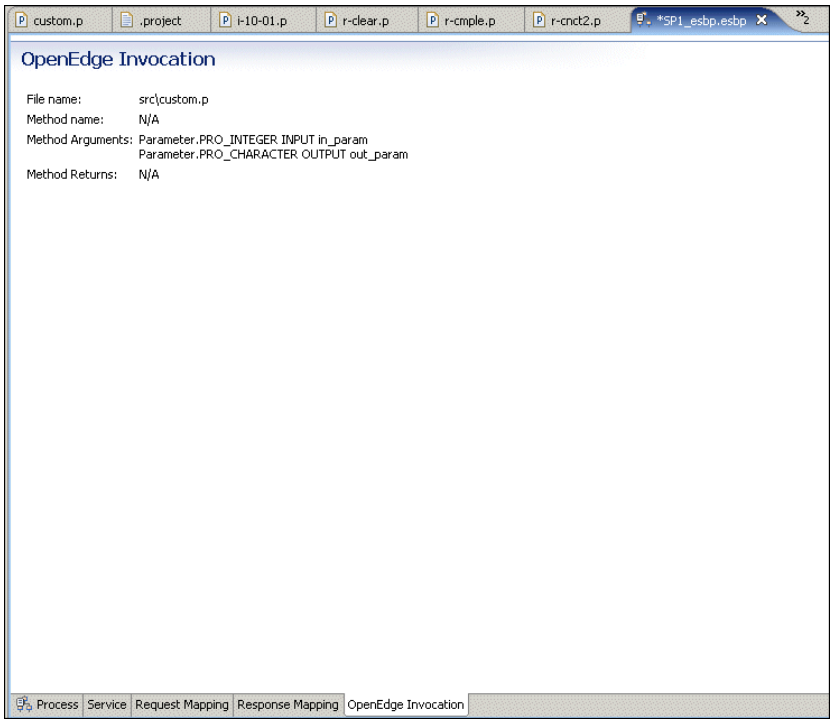
ESB process details

When viewing or developing your ESB process, double-click your ABL procedure to view additional information. The following figure shows the **Service** tab.



When you drag an ABL procedure file into your ESB process, the **Invocation** type is **Inline OpenEdge Invocation**. Clicking **OpenEdge Invocation** displays the invocation details:

Figure 2: OpenEdge Invocation detail



ABL file requirements

ABL files that can be dragged onto the ESB process editor must adhere to the following rules:

- The ABL file must be part of an OpenEdge project. Standalone files cannot be dragged into an ESB process.
- The ABL file must successfully compile. If the file is not compiled when you drag it into the ESB process editor, then it is compiled before the drop completes. If the compile fails, then the file cannot be dropped.
- The file must not be in a modified state. If the file is not saved before you drag it, then it is saved before it is dropped.

In addition, only external procedures, internal procedures, and user-defined functions can be included in an ESB process itinerary, regardless of whether the ABL file is dragged or an `.esboe` file is imported.

Test an ESB process containing OpenEdge native invocation services

After you creat your ESB process with native invocations, you can test it with the following general steps:

1. Save your ESB process, and upload it to your Sonic Domain.
2. In Progress Explorer, configure and start an PAS for OpenEdge instance, typically `oepas1`, to run your ABL procedures.
3. In the Sonic Management Console, confirm that the PAS for OpenEdge parameters for the default **OpenEdge Native Services** `dev.OpenEdge` service is configured to connect to the PAS for OpenEdge instance you started in the preceding step:

a) Click the **Configure** tab and expand **Services**.

b) Select **OpenEdge Native Services**.

c) Select the `dev.OpenEdge` Service Name.

d) Verify the PAS for OpenEdge parameters in the **Init Parameters** section. If necessary, modify the parameters so that they match the PAS for OpenEdge instance.

4. In Sonic Workbench, click the **Container** view. The default container for native invocations is `dev_OpenEdgeTest`. Select the `dev_OpenEdgeTest` container. Right-click and select **Start**, to start the container.
5. Create and run **Scenarios** that verify the execution of your process, including a successful round-trip to and from PAS for OpenEdge. For more information about creating and running **Scenarios**, and testing your ESB process, see your Sonic Workbench documentation.

Configure Progress Developer Studio for OpenEdge for ESB annotations

By default, Progress Developer Studio for OpenEdge is configured with a standard ESB annotation definition and configured to build ESB invocation files. To verify or change these settings, see the following topics:

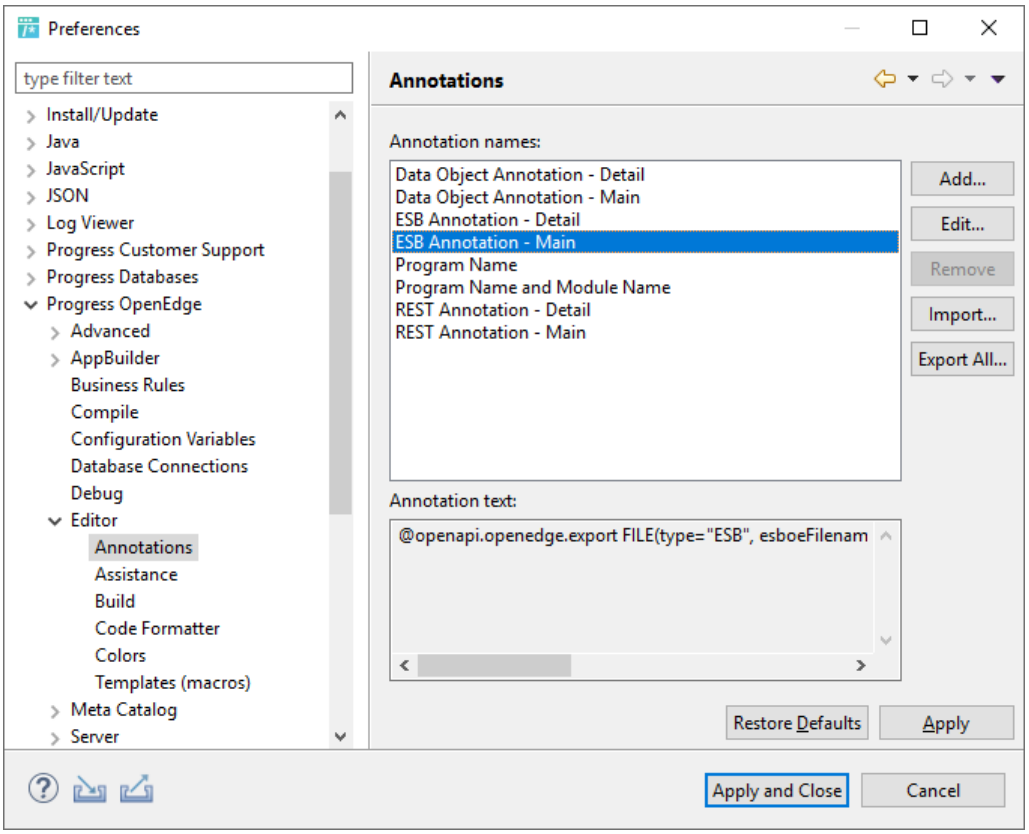
- Default ESB annotations on page 51
- Generate ESB invocation files on page 52

Default ESB annotations

You can verify or change the default Progress Developer Studio for OpenEdge annotations settings.

To verify the Progress Developer Studio for OpenEdge annotation settings:

1. In Progress Developer Studio for OpenEdge, open an existing project or start a new project.
2. Select **Window** > **Preferences**.
3. In the tree view, expand **Progress OpenEdge**, expand **Editor**, and select **Annotations**.
4. Select an ESB annotation from **Annotation names**:



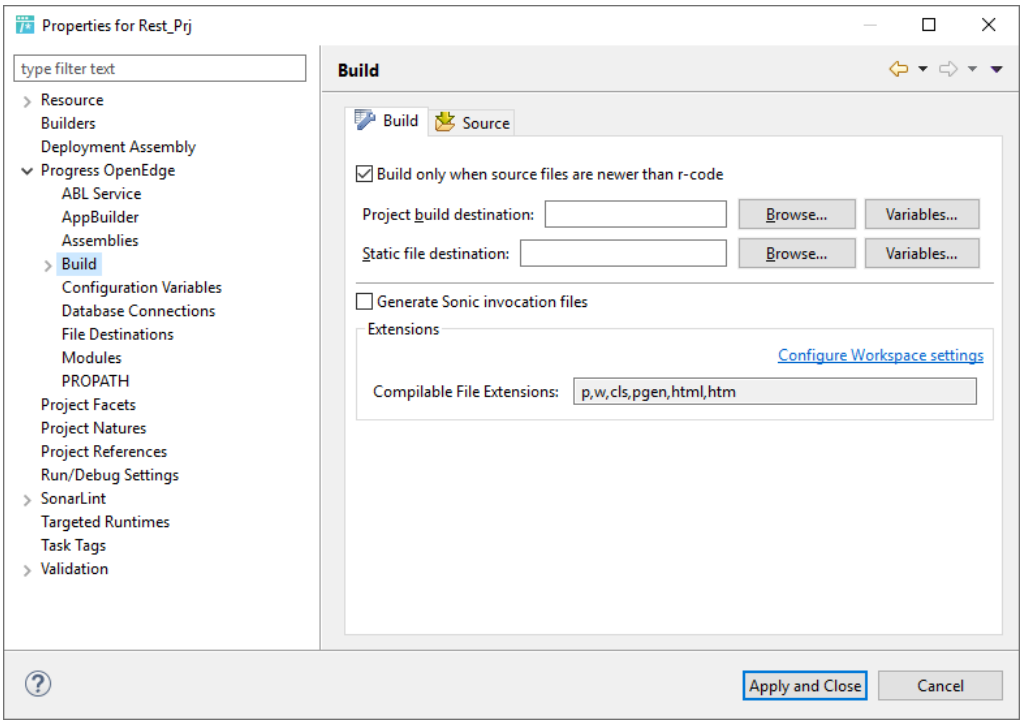
5. Change the default annotation string by selecting **Edit**, if necessary. Click **OK** to exit the **Preferences** dialog box.

Generate ESB invocation files

Progress Developer Studio for OpenEdge can be configured to automatically generate native invocation files when you build your project. You can verify or change the default behavior of Progress Developer Studio for OpenEdge for invocation file generation.

To configure Progress Developer Studio for OpenEdge to generate native invocation files:

1. In Progress Developer Studio for OpenEdge, open an existing project or start a new project.
2. Check your project properties:
   a) Select **Project** > **Properties**.
   b) In the tree view, expand **OpenEdge** and select **Build**.
   c) Select the **Generate Sonic invocation files** checkbox to have Progress Developer Studio for OpenEdge automatically generate your `.esboe` files when your project is built.



Note: By default, Progress Developer Studio for OpenEdge writes your `.esboe` files to your r-code destination directory.

3. Click **Apply and Close**.

The next time your project containing annotated source code is built, Progress Developer Studio for OpenEdge automatically generates your `.esboe` files.

Web Services Invocation methodology

Functionally, there is no difference between the Web Services Invocation methodology in the OpenEdge Adapter for Sonic ESB and the OpenEdge Web Services Adapter (WSA). Both adapters perform the same conversions between the ABL and SOAP protocols. With the Web Services Invocation methodology, an OpenEdge service of Sonic ESB is essentially identical to an OpenEdge web service.

Any service developed for use with the OpenEdge Adapter for Sonic ESB can function as a standard web service, either in the context of Sonic ESB or the WSA. Conversely, any existing web service deployed to a WSA can be exported (through either the Progress Explorer or the WSAMAN utility) and installed into Sonic ESB with its run-time properties intact.

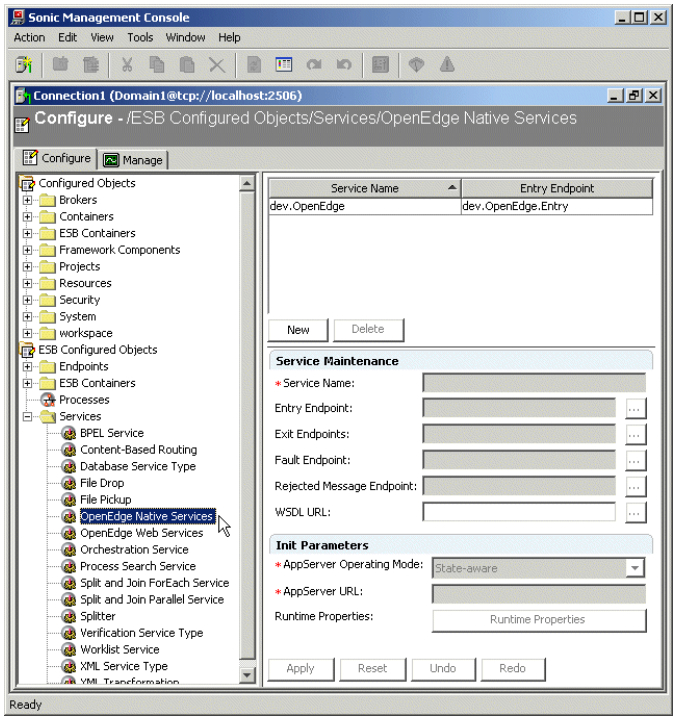
Because of the similarity between the two adapters, the same rules and guidelines apply to programming services and clients.

Differences between the Web Services Invocation methodology in the OpenEdge Adapter for Sonic ESB and the WSA

Although the OpenEdge Adapter for Sonic ESB and the WSA are very similar, there are some differences between them. The most significant difference is that Sonic ESB hosts a service in conjunction with the OpenEdge Adapter for Sonic ESB, whereas an OpenEdge web service is hosted on a web server or Java Servlet Engine (JSE) in conjunction with the WSA.

It is important to note that once the OpenEdge Adapter for Sonic ESB is installed, it does not appear as an entity of that name in any OpenEdge or Sonic UI component. In the **Sonic Management Console** (the primary tool for managing services and related functions) the presence of two OpenEdge services, **OpenEdge Native Services** and **OpenEdge Web Services**.

The following figure shows the OpenEdge services in the **Sonic Management Console**.



There are a few additional differences, described in the topics that follow, that you should note if you are familiar with the WSA. The procedures to which these sections refer, as well as other important information about management of the adapters, is described in *Manage Progress Application Server (PAS) for OpenEdge*.

Service deployment and management

For a Sonic ESB service, after the PAS for OpenEdge application is developed and its service definition is generated, all deployment, configuration, and management activities associated with the service take place in the Sonic environment.

For a WSA-based service, you use the Progress Explorer or the WSAMAN utility to perform deployment and administrative functions.

WSM and WSD file usage

When you install a Web Services Invocation OpenEdge service in Sonic ESB, you specify one of the following two files as the service definition:

- **The WSM file** — Created by ProxyGen from the compiled r-code for a PAS for OpenEdge application
- **The WSD file** — Exported from an existing web service

Sonic ESB stores a copy of the specified service definition file as a resource. The file retains its original name. The OpenEdge Adapter for Sonic ESB relies on the stored WSM or WSD file for the information needed to code and decode client SOAP messages and to access the appropriate service to execute requests.

When you deploy a web service to a WSA instance, in contrast, the WSA stores the service definition as a copy of the original WSM or WSD file with the name *`FriendlyName`*`.wsad`. The WSA relies on this WSAD file for the necessary information about SOAP processing and service identification.

WSAD files do not exist in Sonic ESB.

Storage of property information

Default run-time properties of OpenEdge services for Sonic ESB are hardcoded in the OpenEdge Adapter for Sonic ESB. To change these properties for a given service, you edit its associated WSM or WSD service definition file, which is stored as a resource in Sonic ESB as mentioned in the preceding section. You use the custom OpenEdge Resource Editor for this purpose.

In the WSA environment, default run-time properties are stored in the `default.props` file associated with each WSA instance. You can modify these properties for a given service by using the Progress Explorer or the WSAMAN utility. Each service's run-time properties are stored in a file named *`FriendlyName`*`.props`.

WSDL file generation

When your Sonic ESB service is ready for deployment, you use the custom OpenEdge Resource Editor available in the Sonic ESB Explorer to generate the associated WSDL file that defines the client interface to the service. This procedure differs from the corresponding procedure for WSA-based services in that the WSDL file for those services is generated through the Progress Explorer.

Service definition considerations

The topics that follow contain guidelines for using the ProxyGen utility to create a service definition. For detailed information about ProxyGen, see *Get started with ProxyGen*.

Session models

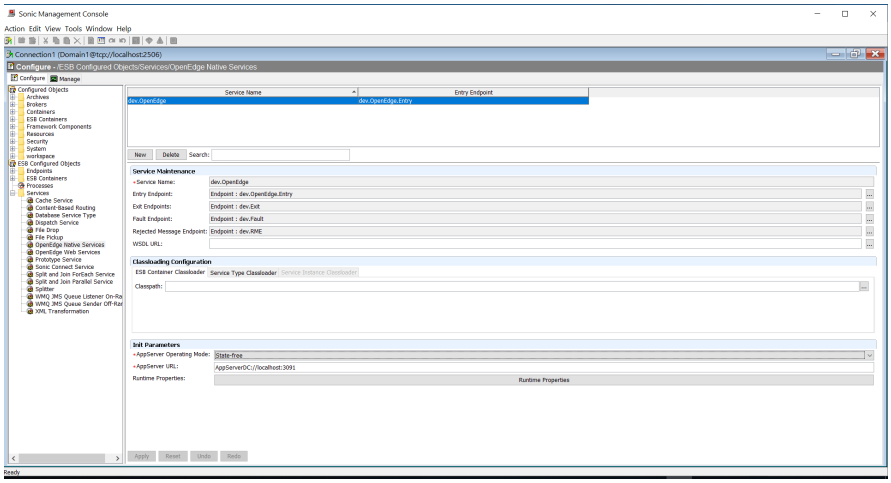
The OpenEdge Adapter for Sonic ESB, like the WSA, supports two session models:

- **Session-managed**—While a transaction is in progress between the service and a client, the connected PAS for OpenEdge instance is dedicated exclusively to that client. The PAS for OpenEdge maintains the context of the transaction until it responds to all requests from the client and the transaction is complete. The session-managed model is available for the stateless, state-aware, and state-reset PAS for OpenEdge operating modes, and it can be used with all Open Client types (.NET and Java clients as well as web service clients) and OpenEdge clients.

- **Session-free**—PAS for OpenEdge does not maintain a transaction context. PAS for OpenEdge returns a complete result in response to each single request, and it does so without regard to any previous responses to the client from itself or any other PAS for OpenEdge process. Any available PAS for OpenEdge processes launched by qualified PAS for OpenEdge broker can process requests from the client in parallel, because each transaction is independent of all others. The session-free model is available on all clients, and it requires PAS for OpenEdge to run in state-free mode.

When you program services that are managed by the OpenEdge Adapter for Sonic ESB and their clients, it is preferable to write applications to take advantage of the session-free model. By doing so, you avoid the need to write code to manage connections. Similarly, it is best to avoid including `SubAppObjects` or `ProcObjects` (persistent procedures) when generating the client proxy for the PAS for OpenEdge application, because such objects are processed in a session-managed mode even if the session-free model is specified for the top-level `AppObject`. In general, the session-free model without the use of subobjects requires less complex programming. It also allows for better performance and scalability, because multiple PAS for OpenEdge instances and processes can be made available to handle requests in parallel.

The following figure shows the **Proxy Generator** window and the **Progress Explorer** window where you set the applicable options for the session model and PAS for OpenEdge operating mode.



Sonic message handling run-time parameters

All messages in Sonic ESB are multi-part messages. Currently, the OpenEdge Adapter for Sonic ESB looks for its input message in the first part, and creates a new message with its output. However, there are cases when you want to have flexibility in how messages are handled. You can have this flexibility by using service run-time parameters. The following run-time parameters let you specify an input message part and an output message part:

- **`InputMessagePart`**—An integer parameter that specifies which message part to use for the input to the OpenEdge service. The message part must be the proper XML that the service is expecting. There is no difference in functionality when using a value of 0 or 1. They reflect the current functionality of using the first part for the input message. The default is 0.
- **`OutputMessagePart`**—An integer parameter that specifies which message part to use for the output of the service call. A value of 0 (the default) retains the current functionality by returning a new message. A value of 1 retains the original message and replaces the first part with the output. This is useful if you want to retain the message properties of the original message.

You set the values of run-time parameters when you create a process. The values are evaluated when the service is invoked. If you specify an output message part, then the original message is kept intact, with the output of the service call placed in the specified part. By default, there is no change in functionality if you do not specify these run-time parameters.

For more information about Sonic run-time parameters, see the Sonic documentation.

4

Run a sample Sonic ESB Process using Native Invocation

This topic describes the creation and testing of a simple ESB process using the Native Invocation methodology. In this example, the ABL procedure `GetCustName.p` is called from an ESB process, providing a customer number as input, and returning the customer name retrieved from the sports2000 database.

This example relies on the default settings. See your Sonic documentation for information about creating more complex ESB processes that do not rely on defaults.

For details, see the following topics:

- Develop ABL source
- Configure OpenEdge servers
- Annotate the source with Progress Developer Studio for OpenEdge
- Build Native Invocation and r-code files
- Prerequisites to start Sonic Workbench
- Create an ESB process in Sonic Workbench
- Update the Sonic Management Console
- Test the ESB process

Develop ABL source

The first step in developing this example is to write the ABL procedure `GetCustName.p`. The procedure requires an `INTEGER` parameter as input and a `CHARACTER` parameter as output. The code is as follows:

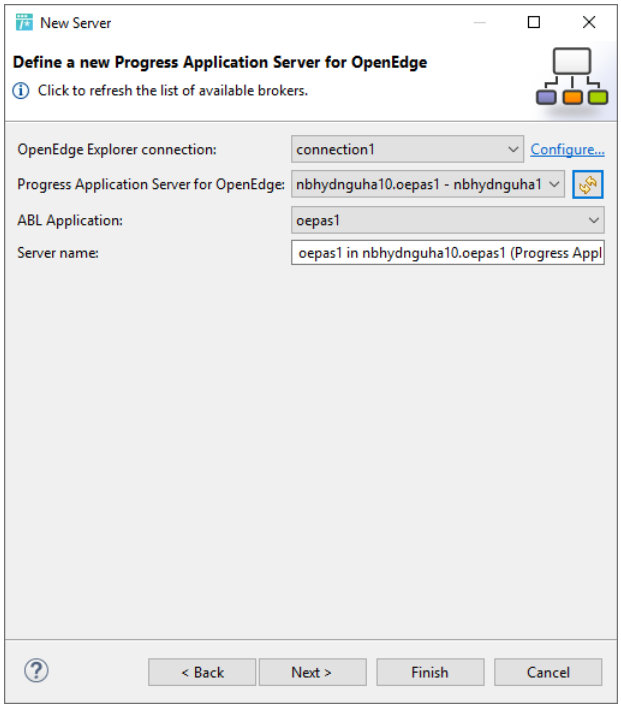
```
DEF INPUT PARAM customerNumber AS INTEGER.
DEF OUTPUT PARAM customerName AS CHAR.
IF CONNECTED("sports2000") THEN DO:
   FIND FIRST customer WHERE custNum = customerNumber NO-LOCK NO-ERROR.
   IF AVAILABLE customer THEN
      customerName = Name.
   ELSE
       customerName = "No record".
   MESSAGE "CustomerName = " customerName SKIP.
END.
```

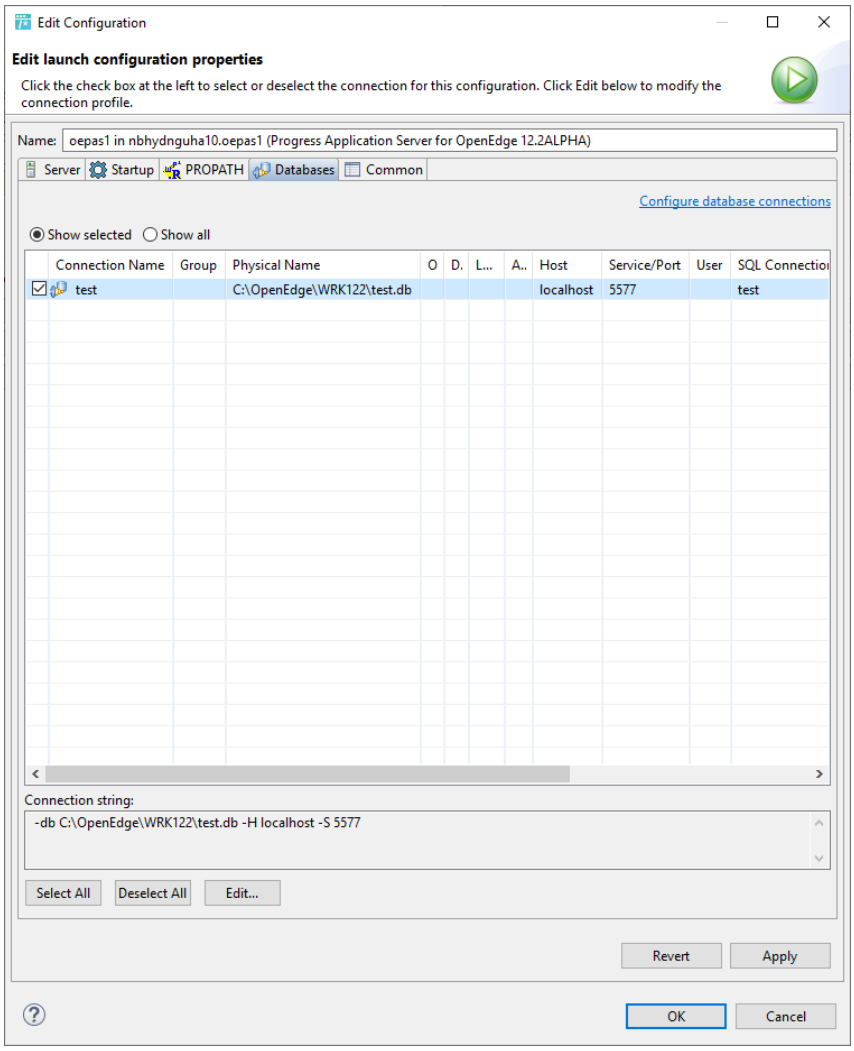
Save the `.p` file.

Configure OpenEdge servers

A successful compilation of your source requires a database connection. Successful testing of your ESB process requires a PAS for OpenEdge connection and a database connection. Start the database and PAS for OpenEdge using Progress Developer Studio for OpenEdge.

Start the database first, and then configure your PAS for OpenEdge instance, `oepas1`, to connect its agents to the database:





After you configure PAS for OpenEdge, start `oepas1`.

Annotate the source with Progress Developer Studio for OpenEdge

Annotate your source in Progress Developer Studio for OpenEdge to provide the information required to create your native invocation file.

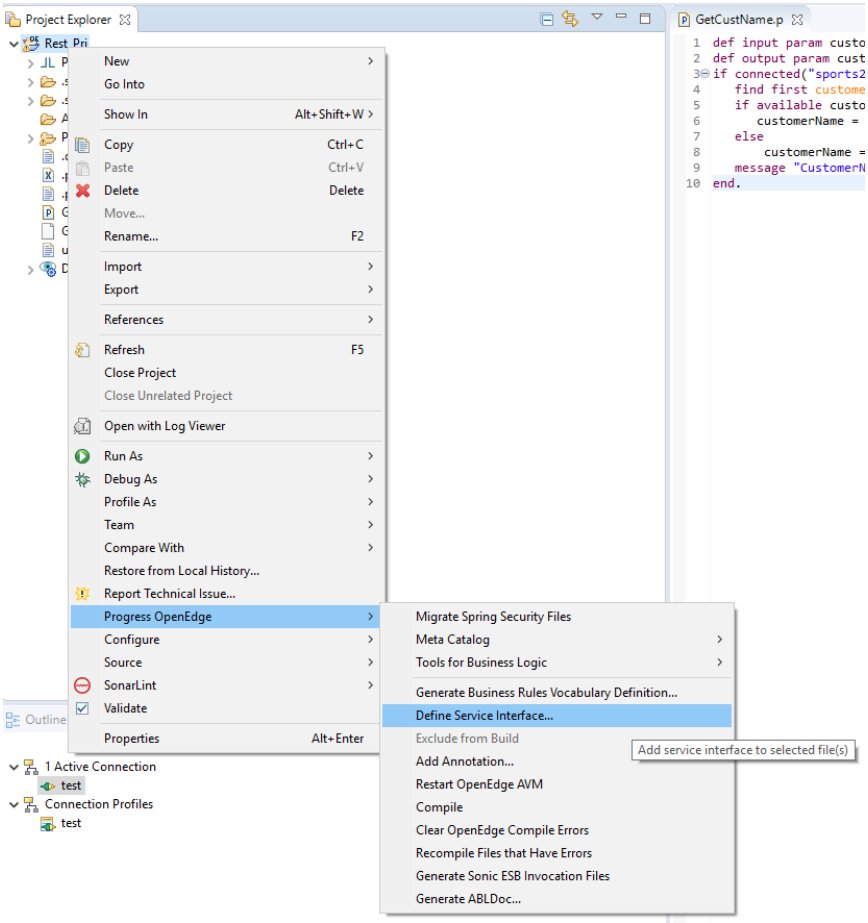
Note: There are other methods to create native invocation files, such as ProxyGen, but creating them from the annotated source is the preferred method. For more information on ProxyGen, see *Use the Open Client Toolkit*.

To annotate your source:

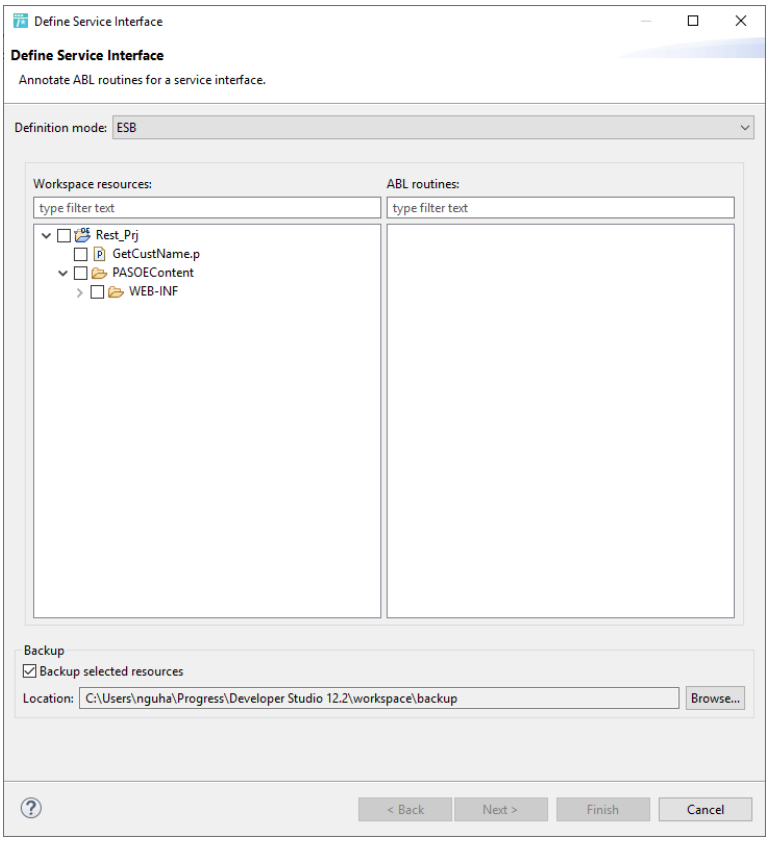
1. Start Progress Developer Studio for OpenEdge and open your source file.

Note: Your source file must be in a project. For information about projects, see: *Progress Developer Studio for OpenEdge Online Help*.

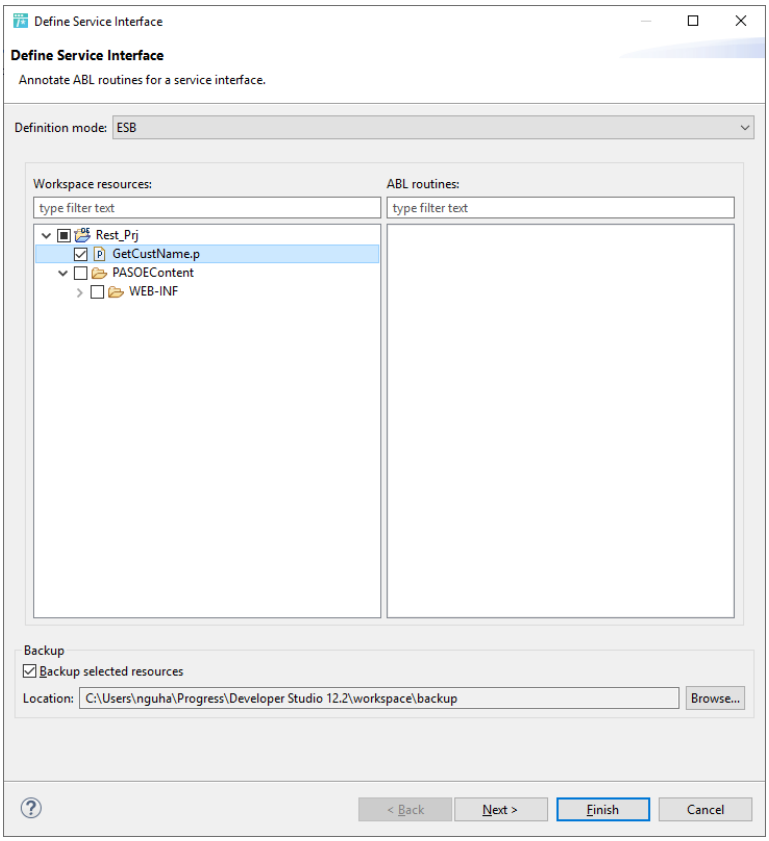
2. Right-click the project, and select **Progress OpenEdge** > **Define Service Interface**. The **Define Service Interface** dialog box appears.



3. Select **ESB** from the **Definition mode** drop-down list.



4. Select the procedure file and click **Finish**.



The following is the annotated source:

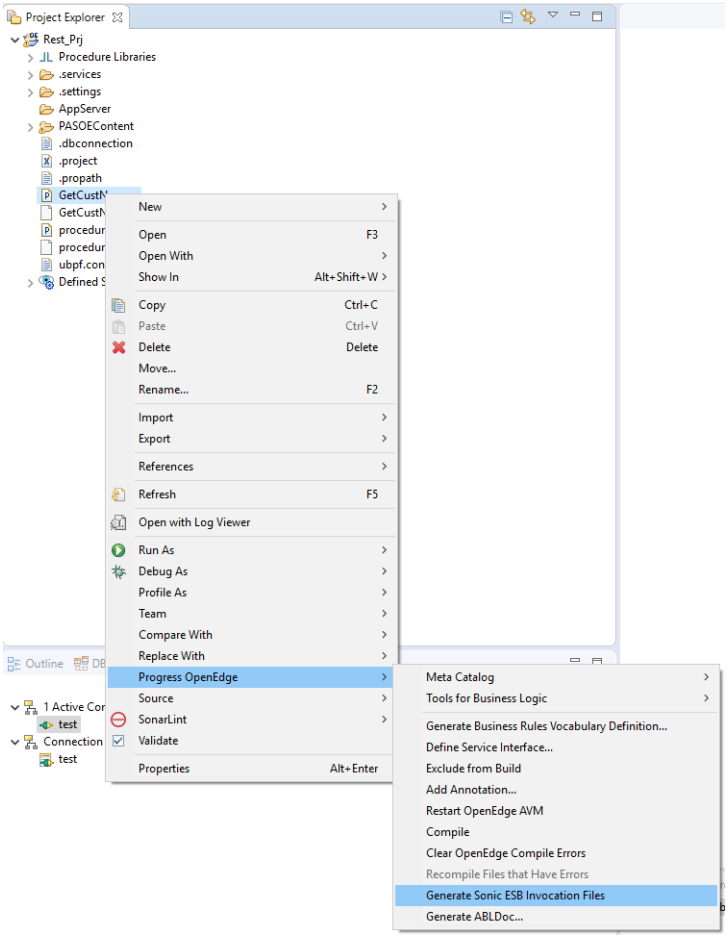
```
@openapi.openedge.export FILE(type="ESB", esboeFilename="%FILENAME%",
useReturnValue="false", writeDataSetBeforeImage="false", executionMode="external").
DEF INPUT PARAM customerNumber AS INTEGER.
DEF OUTPUT PARAM customerName AS CHAR.
IF CONNECTED("sports2000") THEN DO:
   FIND FIRST customer WHERE custNum = customerNumber NO-LOCK NO-ERROR.
   IF AVAILABLE customer THEN
      customerName = Name.
   ELSE
       customerName = "No record".
   MESSAGE "CustomerName = " customerName SKIP.
END.
```

Build Native Invocation and r-code files

After you annotate your source, build your native invocation (`.esboe`) and r-code files. You must connect the **sports2000** database to your project before proceeding. To learn how to connect to a database, see *Connect a database*.

To build the `.esboe` and r-code files:

From the **Project Explorer** tab, select the procedure file. Right-click and select **Progress OpenEdge** > **Generate Sonic ESB Invocation Files**.



OpenEdge compiles your source, saves it to r-code, and generates the native invocation file. Success is indicated by the following output in the **Console** window:

```
>GetCustName (C:\Users\name\Progress\Developer Studio
12.2\workspace\Rest_Prj\GetCustName.esboe)
>GetCustName.p (C:\Users\name\Progress\Developer Studio
12.2\workspace\Rest_Prj\GetCustName.r)
```

Prerequisites to start Sonic Workbench

Before you start and run **Sonic Workbench**:

1. Start **Sonic Domain Manager**.
2. Navigate to `$DLC\sonic\ESB10.0\config` and run `offline.bat` to create the containers.
3. Open **Sonic Management Console** and enter your username and password.
4. Open **Sonic Workbench**.

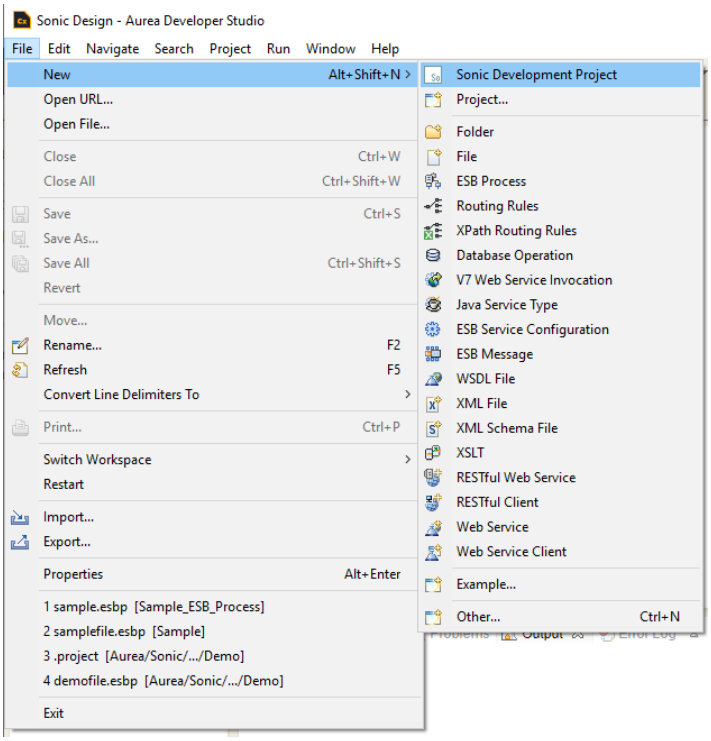
.

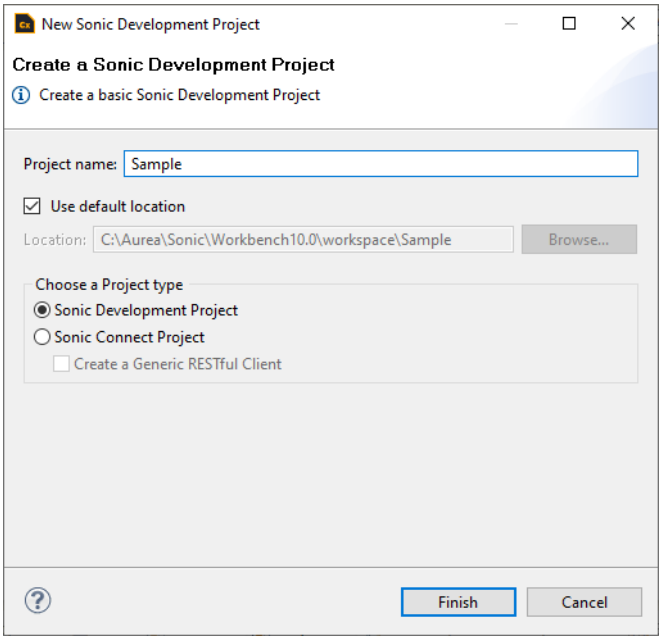
Create an ESB process in Sonic Workbench

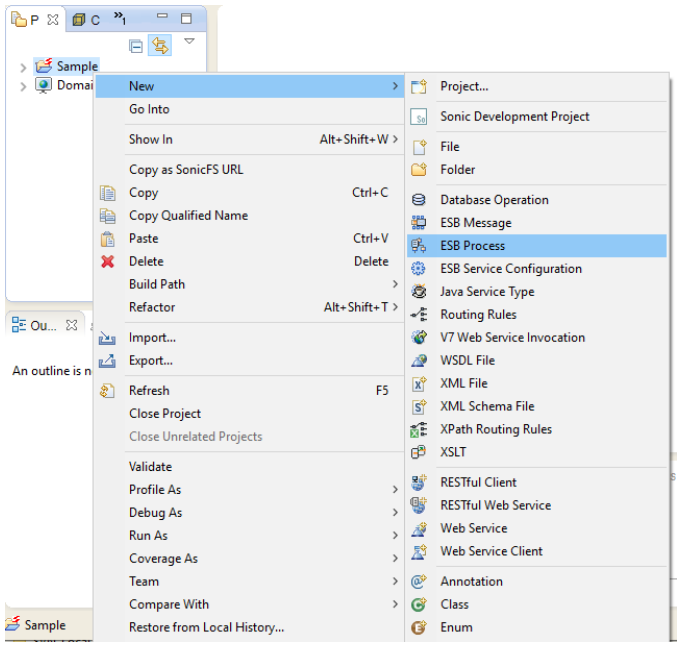
After you you develop your ABL procedure and create a native invocation file, create an ESB process in Sonic workbench.

To create an ESB process in Sonic Workbench:

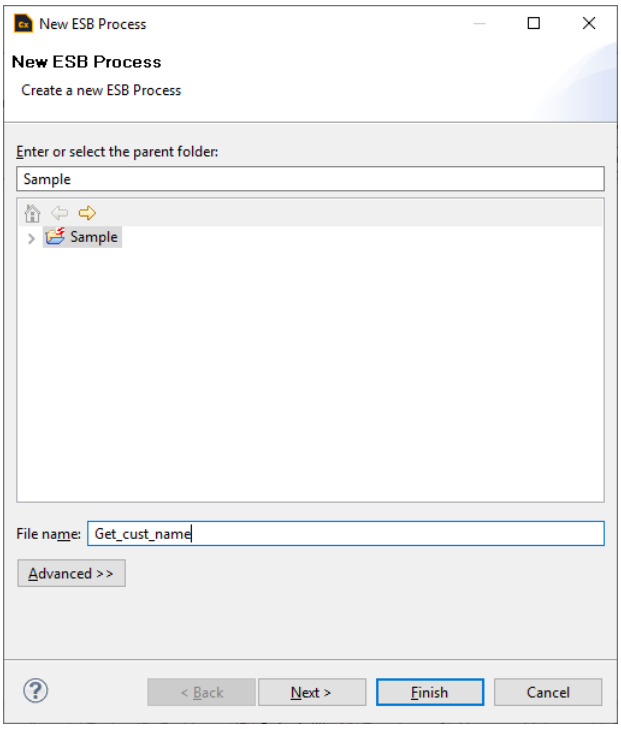
1. Create a new Sonic Development Project, and then create a new **ESB Process**.The **New ESB Process** dialog box opens.

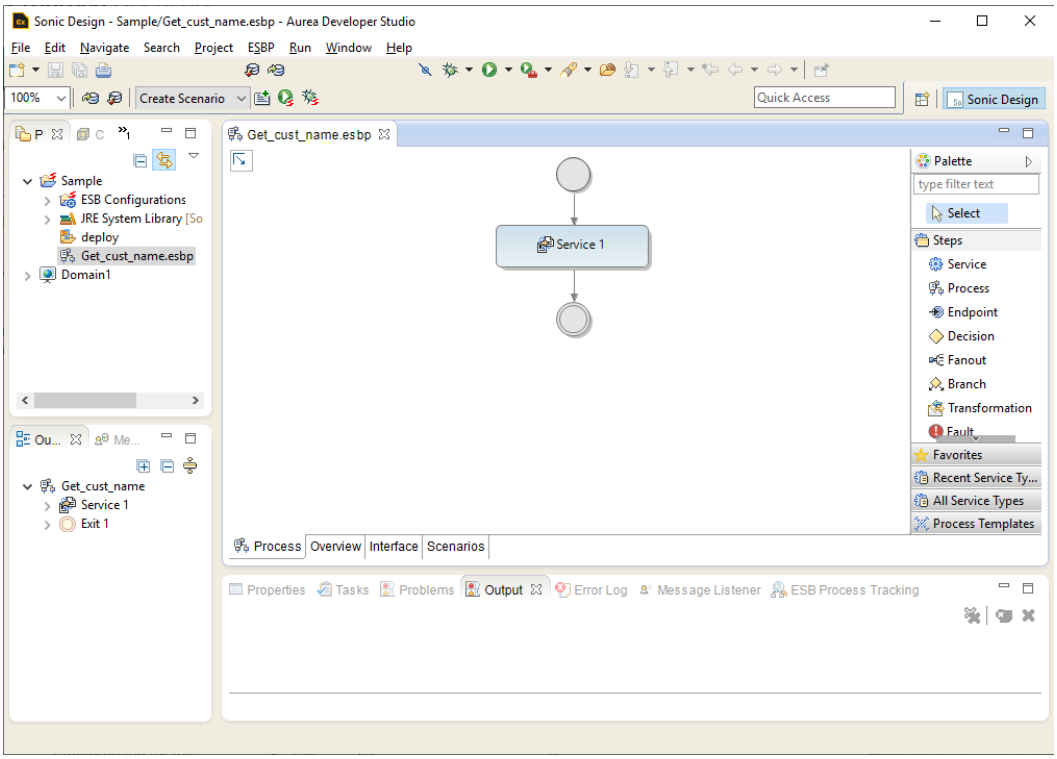

Sonic Design - Aurea Developer Studio
File Edit Navigate Search Project Run Window Help
New Alt+Shift+N
Open URL...
Open File...
Close Ctrl+W
Close All Ctrl+Shift+W
Save Ctrl+S
Save As...
Save All Ctrl+Shift+S
Revert
Move...
Rename... F2
Refresh F5
Convert Line Delimiters To
Print... Ctrl+P
Switch Workspace
Restart
Import...
Export...
Properties Alt+Enter
1 sample.esbp [Sample_ESB_Process]
2 samplefile.esbp [Sample]
3 .project [Aurea/Sonic/.../Demo]
4 demofile.esbp [Aurea/Sonic/.../Demo]
Exit
Sonic Development Project
Project...
Folder
File
ESB Process
Routing Rules
XPath Routing Rules
Database Operation
V7 Web Service Invocation
Java Service Type
ESB Service Configuration
ESB Message
WSDL File
XML File
XML Schema File
XSLT
RESTful Web Service
RESTful Client
Web Service
Web Service Client
Example...
Other... Ctrl+N


New Sonic Development Project
Create a Sonic Development Project
Create a basic Sonic Development Project
Project name: Sample
Use default location
Location: C:\Aurea\Sonic\Workbench10.0\workspace\Sample
Browse...
Choose a Project type
Sonic Development Project
Sonic Connect Project
Create a Generic RESTful Client
Finish
Cancel

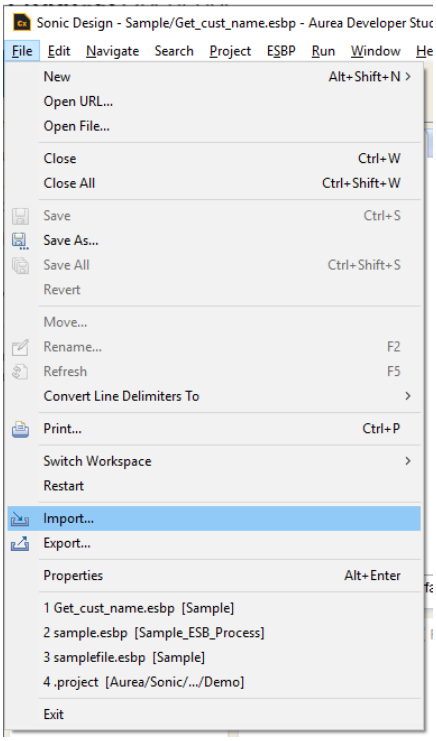


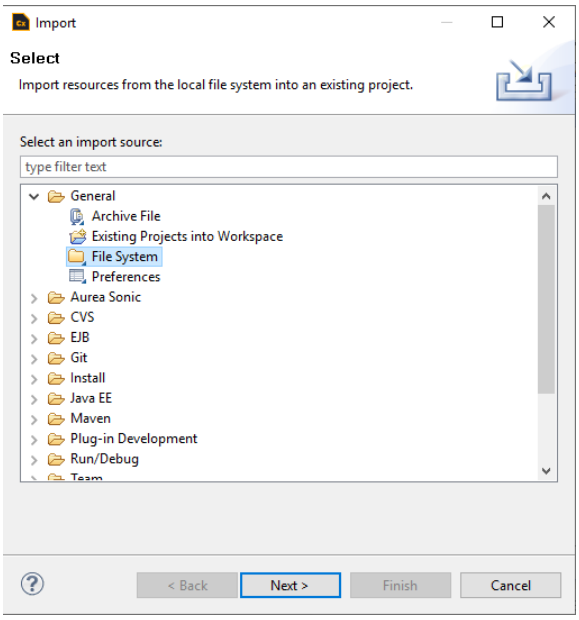
2. Name your process and click **Finish**. An empty process appears:

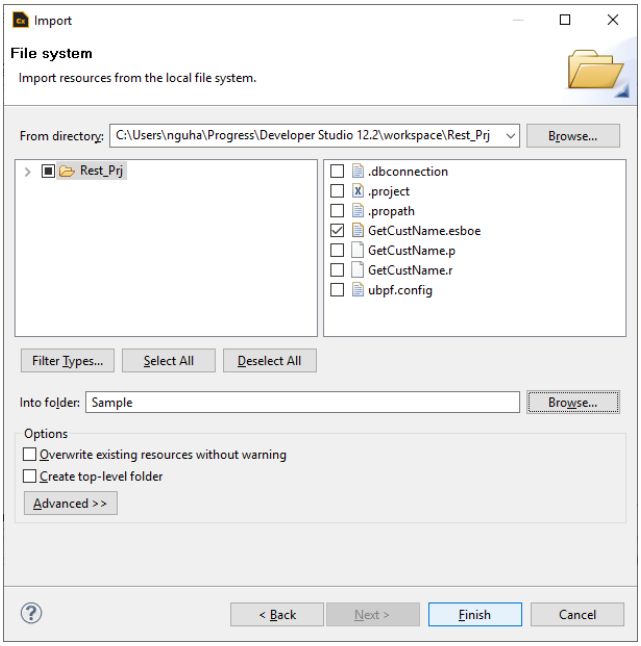




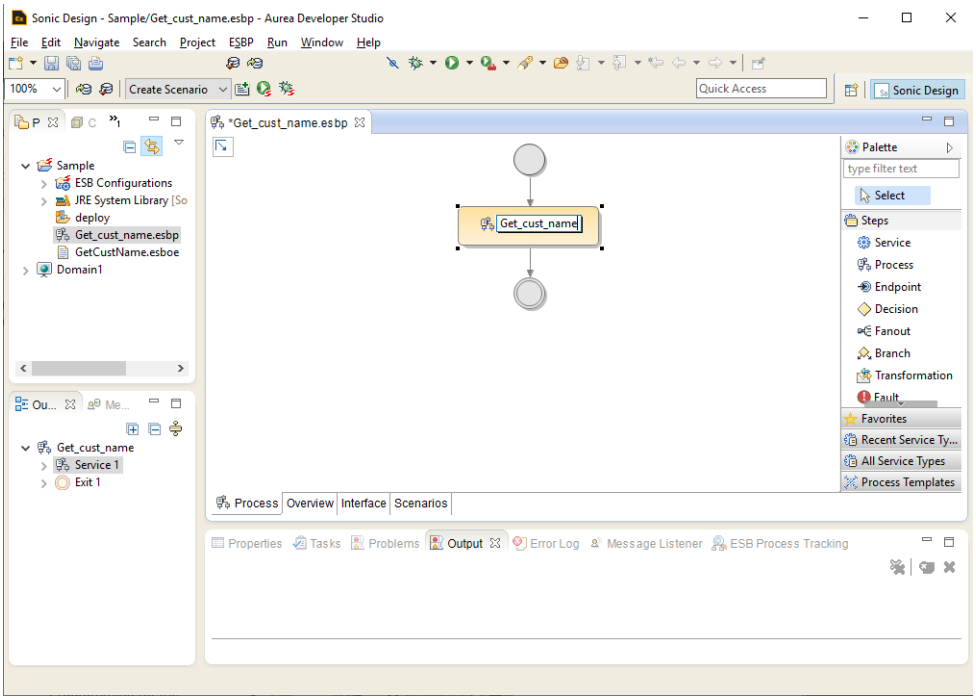
3. To import the invocation file, select **File** > **Import**.
4. In the **Import** dialog box, select **File System** from the **General** directory and click **Next**.
5. Browse to the directory where your native invocation file is located, select your invocation file, and specify the location of your project in the **Into folder** field. Click **Finish**.



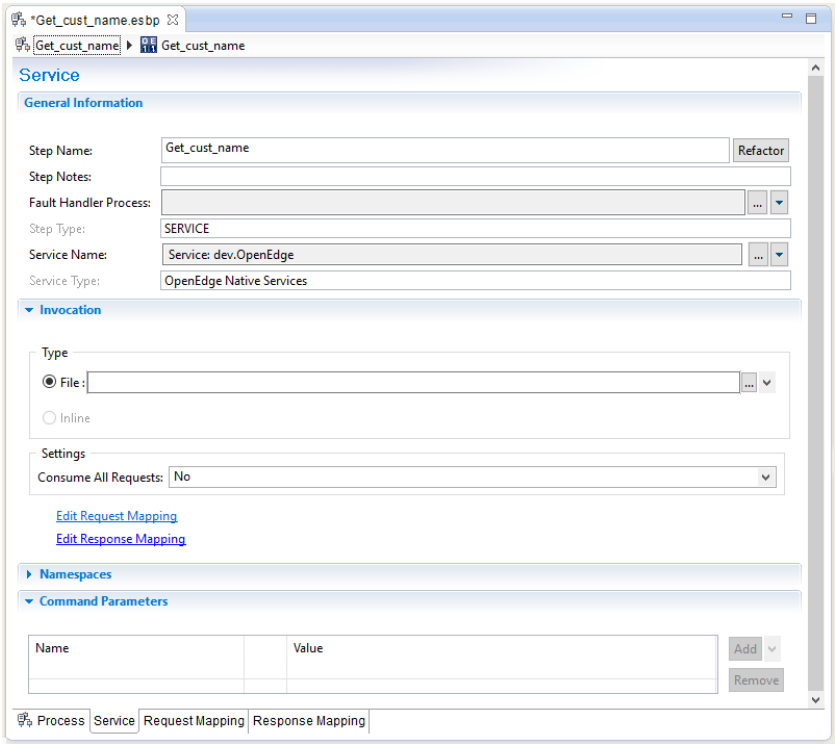




6. Add the **Native Invocation File (GetCustName.esboe)** to the process by dragging the file from the **Project Explorer** window onto the process.

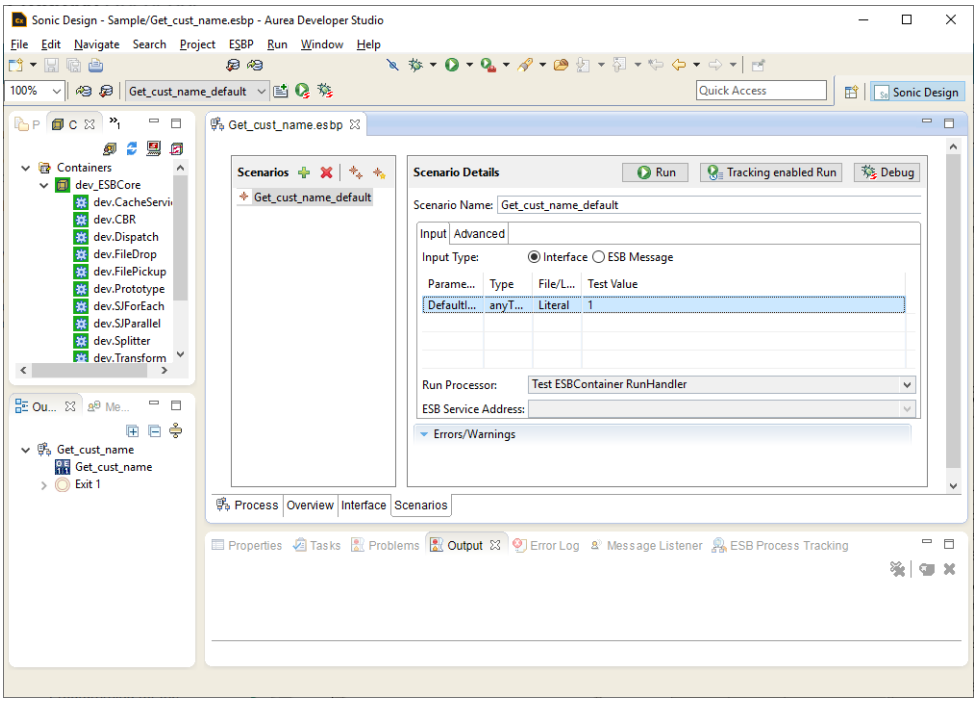


a) If you want to examine the process details, then right-click the service and select **Open**. The details appear, as shown:



7. To add a scenario:
   - Click the **Scenario** tab.
   - Click **+** to add scenario.
   - In the **Scenario Details** section, change the **File/Literal** column to **Literal** .

- Provide a **Test Value**.



8. Save your process, and click **Yes** when you are prompted to upload the process after saving.

Update the Sonic Management Console

After you create an ESB process and update the scenario in Sonic Workbench, you can update the **PAS for OpenEdge** details in the **Sonic Management Console**.

To update the PAS for OpenEdge details in the Sonic Management Console:

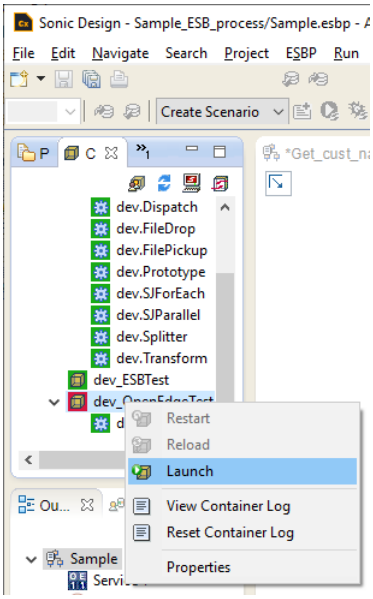
1. Open the Sonic Management Console and in the **Configure** tab double-click the **Services** directory.
2. Select **OpenEdge Native Services**.
3. Select `dev.OpenEdgeTest` Native Service in the **Service Name** section.
4. In the **Init Parameters** section, update the PAS for OpenEdge details as follows:
   - **AppServer Operating Mode** as **State-Free**.
   - **AppServer URL** as *http://localhost:8810*/apsv.
5. Click **Apply**.

Test the ESB process

Your example process is complete and ready to be tested.

To test your ESB process:

1. In **Sonic Workbench**, switch to the **Container** tab, select the **dev_OpenEdgeTest** container, right-click the container and select **Launch**.



2. Open the `openedge.properties` file from the instance's working directory. In this example, it is in *`C:\OpenEdge\WRK122\oepas1\conf`*.
3. In the properties file, update the `agentStartupParam` details as `agentStartupParam -db` *`database name`* `-H localhost –S` *`port number`*.
4. Place a copy of the procedure file (`GetCustName.p`) in the instance's working directory (*`C:\OpenEdge\WRK122\oepas1\openedge`*).
5. Go back to **Sonic Workbench**.
6. Run the scenario by clicking the **Run** button in the scenario details section. The results of running the scenario appear in the **Output** window:

In this scenario, a value of '1' was sent to the ABL procedure `GetCustName`, and it returned the customer name, demonstrating a successful execution of the ABL procedure by the Sonic ESB process.